Pothole Detection and Tracking
Midterm Document
University of Central Florida
Dr. Chung Yong Chan
Group 2

Authors:

Travis Grant - Computer Engineering
John Billeci - Photonic Science and Engineering
Samuel Welch - Electrical Engineering
Jose Kostyun - Computer Engineering
Brandon Skervin - Mechanical Engineering

Reviewer Committee

Dr. Peter Delfyett	CREOL	Committee
Dr. Arup Guha	CS	Committee
Dr. Mark Maddox	ECE	Committee
Dr. Sudeshna Pal	MAE	Committee

Mentor: Dr. Chung Yong Chan

Table of Contents

2. Project Description

2.1 Motivation and Background

Maintaining safe roads is a constant challenge for city governments, particularly when operating with limited budgets. One of the most widespread issues is potholes, which can cause vehicle damage, increase accident risk, and put a strain on municipal maintenance operations. Detecting and tracking these hazards quickly and accurately is essential to ensuring public safety. However, current methods rely on manual surveys, inconsistent user reports, or expensive data collection.

Our project, the Pothole Detection and Tracking System, aims to automate this process with a cost-effective, vehicle-mounted solution that simplifies the gathering, verification, and reporting of pothole data. This system uses infrared light to cast a structured light pattern onto the road surface behind a moving vehicle. A filtered, wide-angle infrared camera captures reflections of this light, and deviations in the laser pattern indicate the presence of a pothole or surface anomaly. The captured IR data is processed in real time by an onboard Raspberry Pi, which runs a laser line deviation detection algorithm.

If a potential pothole is detected, the system triggers AI-based image confirmation using buffered RGB frames from a separate visible-light camera. A lightweight YOLOv5n model verifies whether the anomaly is a true pothole. The system also includes an MSP430 microcontroller on the custom PCB, which handles low-level control tasks such as laser driver management and UART communication with the Raspberry Pi. Confirmed detections, along with measurement data, are sent via Bluetooth to a paired smartphone. The app logs the GPS coordinates of the pothole and uploads the data to a cloud-based server for future analysis.

The system is designed with accessibility and real-world use in mind. A custom rear-mounted bracket allows for both temporary and permanent attachment to almost any standard vehicle. Power can be drawn from a vehicle's battery system or an independent power supply if needed. We chose to keep the total cost of components below $1,200 to ensure the device remains affordable for widespread development by local governments and smaller contractors.

Unlike more expensive systems, such as vehicle-integrated road monitoring offered by manufacturers or LiDAR-based platforms like XenoTrack, our design provides a lightweight, standalone option that customers can adopt without overhauling existing infrastructure. Compared to user-driven apps like Waze or post-processed video analysis tools such as RoadBotics, our project offers consistent, autonomous detection with real-time reporting and built-in location services.

Initial development has focused on creating a functional prototype that includes an IR laser and camera setup, embedded software for pattern detection, IOS application, and secure Bluetooth communication. Future plans for this project include expanding the app to Android and enabling more advanced features such as depth estimation and volume analysis for potholes. We also plan to improve the adjustability and durability of the mounting system to support a wider range of road and weather conditions.

Ultimately, the Pothole Detection and Tracking System uses practical innovation to help solve an issue that affects many people on a daily basis. By streamlining the pothole detection process, our system offers realistic, low-cost tools that give cities and customers the data they need to take quick action in repairing potholes and improving roadway safety.

2.2 Features and Functionalities

The Pothole Detection and Tracking System is designed to achieve the following:

Bluetooth Connection: The Raspberry Pi should be capable of communicating with the app on the user's phone using Bluetooth 5.0 technology. It should be able to pass the width of the pothole, and possibly an infrared image for human confirmation.

IOS/Android App: The data will be accessible using an app on the user's phone. This will show a list of potholes in their area. The app will also feature a map with pinned locations, representing detected potholes. When a detection is made, the app will use the phone's GPS to mark the location of the pothole.

Custom Mounting: The Sensors will use a custom mounting device that will be compatible with all vehicles. The device will have multiple degrees of freedom, allowing it to get the right height, angle, and distance from the vehicle. This device should also have the ability to use magnets, suction cups, and screws as options for mounting.

IR Laser Sensor: Part of the sensor will include a visible camera with a near-infrared bandpass filter that runs at 90 frames per second at 480p. The Laser part of the sensor will cover at least a 6 feet by 2 feet area, with the ideal size being a 10 feet by 2.5 feet area.

AI Image Detection: A separate RGB visible-light camera will be used to capture higher-resolution images, which are stored in a buffer. These buffered images are processed through a custom-trained YOLOv5n model to confirm potential potholes detected by the IR laser system before any data is sent over Bluetooth.

Cloud Synchronization and Data Export: The mobile app will feature cloud synchronization, allowing all verified pothole detections to be uploaded to a

database in real time. This ensures that the data can be accessed remotely, is backed up, and is able to be shared among users.

Weather Resilience: This system uses water-resistant housing units, an IR laser, and a filter-based imaging system, meaning our device is less affected by rain and poor lighting. This offers greater reliability than other systems.

2.2.2 Optical Schematic

The Optical Schematic better shows the relationship between the optical/laser systems and the relevant hardware systems of the project. The optical system uses an IR laser to project a sheet of light onto the road surface and a modified visible camera to locate any discrepancies on the road surface by monitoring the integrity of the sheet of light. The camera will need to operate at 82 fps to capture a frame 6 inches apart while moving at 30 mph, if possible operating at 120 fps will allow us to capture a frame every 6 inches while at a max speed of 40 mph.

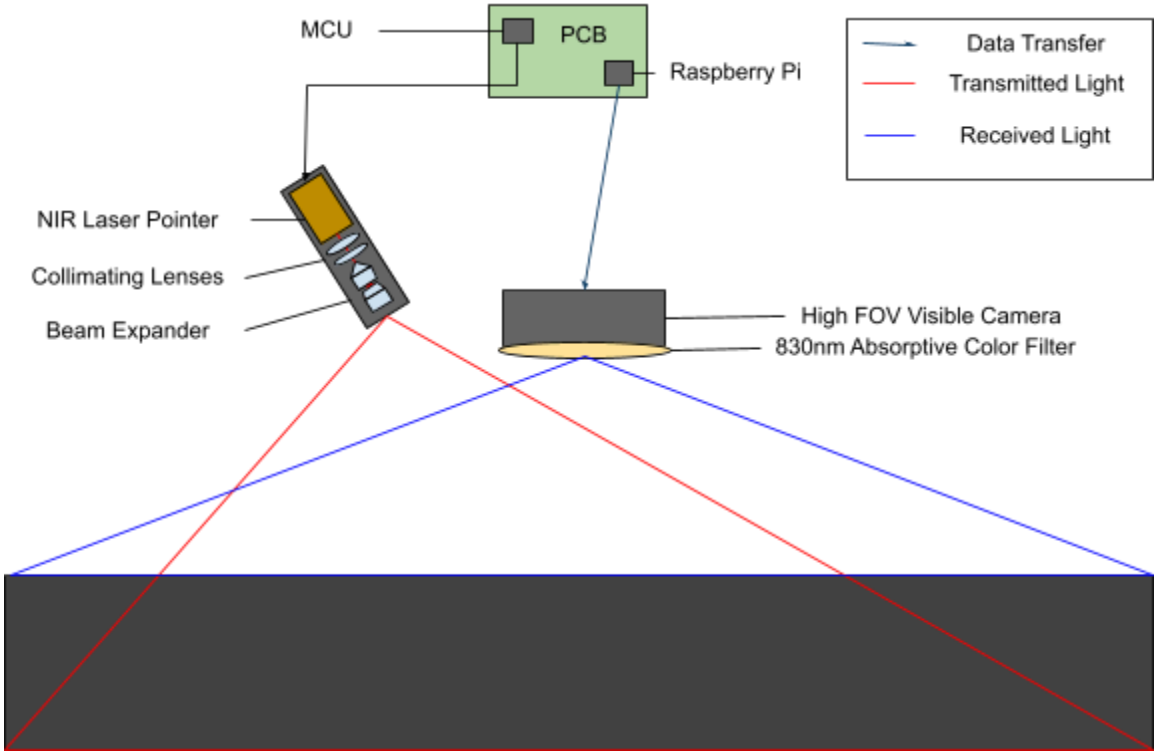


Figure 2.1: *Optical Schematic*

2.2.3 Customer Input:

Cost efficiency: City planners and maintenance departments with limited budgets emphasized the need for a solution that delivers results without the need for high investment. In response, we prioritized the use of low-cost components

and open-source software, ensuring that our system cost was both functional and affordable.

Ease of Installation: Fleet operators and maintenance teams requested a solution that could easily be mounted and removed without the need for specialized tools or permanent vehicle modifications. This led to the development of a custom, non-invasive, mounted bracket compatible with a wide range of vehicle types.

Accurate & Real-time Measurements: Road maintenance teams emphasized the need for prompt and precise detection of potholes. This drove the integration of laser scan image processing, allowing potholes to be accurately detected, with minimal false positives.

2.2.4 Comparable Products/Projects:

AI Pothole Patrol Systems (Chicago, IL, San Jose, CA, Memphis, TN): Several cities have implemented AI and machine learning systems to analyze video footage from city vehicles to detect potholes with high accuracy. For instance, Memphis has equipped public works vehicles with AI-enabled cameras that process footage to identify issues like potholes and trash, achieving a 90% accuracy rate. These systems can automatically generate work tickets, streamlining maintenance operations. Some implementations also integrate LiDAR, allowing for precise measurements of pothole size and depth. These technologies demonstrate the viability of automated pothole detection and inspired our system's fusion of sensors and onboard analysis.

Pros**:** Enhances efficiency in infrastructure maintenance, reduces manual inspections, and enables proactive repairs.

Cons**:** Initial setup and training of AI models require significant resources, and dependence on cloud services may raise data privacy concerns.

Integrated OEM Solutions (Tesla, Mercedes, Ford): Some modern vehicles, like Teslas and newer models from Mercedes and Ford, have built-in systems to detect rough roads and potholes. Tesla uses cameras and sensors to find bad patches and adjust the suspension to reduce damage. Mercedes sends pothole alerts to nearby drivers using car-to-car messaging. Ford uses sensors to adjust the suspension in real time when a wheel hits a pothole. These tools are impressive, but they're locked into expensive vehicles and can't be added to other fleets. Our system, in contrast, is low-cost, works on any vehicle, and is made for city use.

Pros**:** Improves ride comfort and can help avoid damage to vehicles.

Cons**:** Only available in certain cars and can't be used for city-wide road scanning.

XenoTrack: XenoTrack is a LiDAR system that mounts to a car and creates a high-resolution 3D scan of the road. It's very accurate and great for spotting potholes, bumps, and cracks. It's mainly used in big infrastructure projects and has deep integration with mapping tools. The downside is that it's very expensive and designed for large-scale road studies, not day-to-day use by cities. Our system offers similar features using cheaper parts, making it better for local fleets.

Pros**:** Ideal for road surveys and getting detailed road condition maps.

Cons**:** Too expensive for small cities or wide deployment.

RoadBotics: RoadBotics uses a smartphone mounted on the dashboard to record the road and upload video for AI analysis. It can detect surface problems like cracks and potholes using that footage. The benefit is that it's cheap and uses everyday phones. The downside is that the system is not real-time. It has to be uploaded and analyzed later. Our system does all processing on the spot, which gives cities instant feedback.

Pros: Uses common devices and doesn't need special hardware.

Cons: Can't detect in real-time and needs cloud-based processing.

Waze: Waze lets drivers report road problems like potholes through the app. Cities can partner with Waze to access this data to help plan repairs. While this is a free and useful system, it depends on people taking time to report issues, which leads to inconsistent results. Our design solves this by automating the detection process, meaning every pothole gets logged whether a person sees it or not.

Pros: Free to use, no hardware needed, and connects to city planning tools.

Cons: Data depends on user input, which can miss lots of issues.

Imagevision: ImageVision uses AI and forward-facing cameras to detect potholes from video footage. This works well in many cases, but can be affected by bad lighting, glare, or dirt on the lens. Our system uses infrared laser scanning, which isn't as sensitive to lighting conditions. That makes our detection more reliable, no matter the time of day or weather.

Pros: Uses standard cameras, good image analysis tools.

Cons: Needs clean, clear video and ideal lighting to work well.

2.3 Goals and Objectives

The primary aim of this project is to develop a reliable, cost-effective, and intelligent pothole detection system. This system must be capable of scanning

road surfaces from a vehicle, identifying and measuring potholes using laser-based sensing, and logging results with accurate location data. Additional emphasis is placed on real-time data access via mobile applications and robust environmental durability for deployment in varied weather conditions. The goals have been divided into basic, advanced, and stretch categories, reflecting the increasing complexity and technical ambition of each.

Basic Goals

- Scan an area of 2x6 ft from about 4 ft off the ground.
- Have a high SNR with a 50 mW laser diode.

The basic goal of this project is to develop a device that can analyze a road surface area of 2×6 feet from a mounting height of about 4 feet. This area ensures reasonable coverage behind a moving vehicle without requiring excessive mounting hardware. To ensure high-quality signal capture, the system must maintain a high signal-to-noise ratio using a 50 mW laser diode. This ensures the reflected laser signal can be clearly detected by the camera sensor despite the challenges posed by road surfaces and daylight conditions.

- Reliably detect >95% of potholes that are larger than 4 inches in diameter.
- Be able to differentiate between the laser light and noise.
- Block out visible light using a bandpass filter so that a visible camera can capture the NIR laser light.

To eliminate interference from ambient light, particularly sunlight and headlights, a narrow-band absorptive color bandpass filter will be used, allowing only the desired near-infrared laser wavelength to pass through. The system must reliably detect at least 95% of potholes larger than 4 inches in diameter.

- Keep the budget for the entire project under $1,200.
- Draw power from the vehicle that the device is attached to. If this is not an option, create our own power supply that can provide enough power for the device to run for extended periods of time.

The design must be financially viable and remain under a $1,200 development budget. This budget was selected to allow the system to remain inexpensive, while also taking into consideration that all funding is entirely from our own finances. The device will be powered by the vehicle it’s attached to; however, if direct vehicle power is not an option, a custom battery pack or onboard power solution will be developed to keep the system running for extended periods.

- Determine the GPS coordinates of detected potholes with >95% accuracy using the GPS on the user's phone.
- Make data easily accessible from a user’s phone.
- Use AI object detection model to confirm pothole detections.

Pothole location data must be accurate, with GPS coordinates logged using the mobile device's integrated GPS system and displayed within a user-friendly IOS app. Durability is essential, so the sensor housing must withstand varying weather conditions such as rain. A mounting device must also be designed that can universally be fitted onto any vehicle. This mounting device should also reduce road vibrations ensuring smooth sensor readings. Finally, the system will implement a lightweight AI model, such as YOLOv5n, to validate pothole detections, reducing false positives and improving reliability.

- Protect internal electronics from weather elements (water, dust, debris), meeting the IP65 standards.
- Prevent sensor misalignments caused by vibrations and bumps, using proper damping techniques.
- Properly secure the system to vehicle exterior without drilling or vehicle modification, using suction cups.

The mechanical system needs to ensure that all electronics are properly protected from environmental factors such as rain, dust, and road debris. This is accomplished by meeting IP65 enclosure specifications through a weatherproof housing design. Additionally, the mounting framework must minimize vibration frequency from the road to prevent sensor misalignment that would impact detection accuracy. The proper damping materials and structural rigidity are used to keep the correct orientation of the laser and cameras during driving. Finally, to ensure the system can be used universally, the entire system must mount securely to the outside using suction cups. This facilitates easy installation and removal on a wide range of vehicles, while providing structural integrity in the real world driving environment.

Advanced Goals

- Scan an area of 2.5x10 ft from about 4 ft off the ground.
- Reliably detect >95% of potholes that are larger than 2 inches in diameter.

Once basic goals are achieved, the system will be enhanced to increase detection area coverage to 2.5×10 feet, improving the rate at which road conditions are monitored. The detection capability will also be upgraded to identify smaller potholes, those at least 2 inches in diameter, with the same high reliability of >95%.

- Accurately estimate the area of potholes with less than 5% error.
- Be able to estimate the depth of potholes with less than 5% error.
- Detect if a pothole is filled with water and subsequently cancel depth estimations.

In addition to binary detection logic, the system will now attempt to calculate pothole size by estimating both area and depth, with a target accuracy of under 5% error for both. This will enable road maintenance teams to better prioritize repairs. However, if water is detected in the pothole, which could distort depth readings, the system will automatically cancel or flag depth data as invalid to maintain reporting accuracy.

- Design mounting system that avoids resonance within (5-30 Hz) range
- Allow for adjustable mounting points for camera and laser alignment for accurate sensing
- Ensure the system is adjustable in height and reach, allowing for universal use across a variety of vehicle types.
- Maintain internal operating temperature between 0°C and 45°C within the housing unit to ensure reliable performance for onboard electronics.

The mounting system must be designed so that it can avoid resonance within the 5–30 Hz range of vibrations that are normally encountered on the road, to prevent fatigue and sensor misalignment. To ensure peak optical performance, the system must also include adjustable mounting points for the laser and camera modules for precise alignment and calibration. This flexibility ensures that both sensors are properly directed regardless of the mounting angle or vehicle configuration. The mount must also be height adjustable and extendable so it will accommodate a wide variety of vehicles without compromising structural stiffness. Internally, the enclosure must have a stable working temperature range of 0°C to 45°C to allow for the proper working of sensitive components like the IR camera, the RGB camera, and the laser emitter. Thermal management techniques like ventilation are used to attain this level of temperature without using active cooling.

- Create an Android app in addition to the IOS app.
- Allow app users to remove detections if necessary.
- Achieve a high SNR with a prebuilt 5 mW laser
- Have a picture of the pothole sent over Bluetooth.

For broader accessibility, an Android app will be developed alongside the iOS version. The app will allow users to manually delete false detections, or potholes that have been repaired, improving data quality. To reduce system complexity and cost, a prebuilt 5 mW laser diode may be used instead of the 50 mW custom unit, but only if it can deliver acceptable SNR performance. Additionally, the system will include functionality to transmit a visual image of the pothole to the user's phone via Bluetooth for further verification or logging.

Stretch Goals

- Be able to provide an estimate of the volume of the potholes with 90% accuracy.
- Be able to estimate the depth and volume of a pothole full of water with 80% accuracy.
- Develop a website in addition to the IOS and Android apps.

Stretch goals are ambitious and will be pursued as time and resources permit. Volume estimation represents a significant leap in data value for repair crews. The goal is to estimate pothole volume with 90% accuracy, providing direct insight into material requirements and urgency. Achieving reliable depth and volume estimation even for water-filled potholes will involve advanced image processing and potentially depth estimation techniques. A target of 80% accuracy will be set for such challenging detections. Finally, a web-based dashboard will be developed to provide centralized access to pothole data. This will allow road maintenance authorities, researchers, or even community members to view and export condition reports from anywhere, expanding the system's reach beyond mobile platforms.

Objectives

1. Determine the ideal method to create a sheet of laser light on the road, taking into account the cost and ability to cover enough of the road from a reasonable height.
2. Expand a beam of laser light using two perpendicular Powell lenses.

3. Take advantage of absorptive color bandpass filters to get a high FOV from the camera while minimizing blue shift.

4. Determine the best method for identifying a disturbance in the laser light for identification purposes.

5. Identify the ideal method of sending information from the device to a cloud server, even in somewhat remote areas.

6. Design a shock-resistant and weatherproof housing for the device to protect the electronics from the elements and prevent any components from being dislodged.

7. Create an IOS app that allows the user to access the pothole location and size data from their phone. This app should have the ability to show active potholes as pin markers on a map. This app uses the phone's location.

8. Develop a system that transmits the stored information when connected via Bluetooth.

9. Develop a cloud server for storing pothole information data.

10. Develop a system to draw power from the vehicle to which the device is attached. If this is not feasible, we will develop a power supply that can power the device for an extended period of time.

11. Create a custom mount for the sensors. It should be able to mount onto any vehicle's rear, either temporarily (using suction cups or magnets) or permanently mounted to the vehicle. The arm should have multiple degrees of freedom, allowing the sensors to be at the right height, angle, and distance from the vehicle.

12. Use YOLOv5n to confirm detected potholes.

13. Give app users with administrative access the ability to review images and confirm or delete detections.

14. Make use of a 50 mW Laser Diode to keep SNR high; if possible, change to a prebuilt 5 mW Laser to decrease complexity and price.

2.4 Specifications

Table 2.4.1: System Specifications

Parameters	Specifications	Priority
Operating Temperature	0℃ to 48℃	High
Pothole Location Accuracy	Within 20 feet	High
Power Consumption	12 V	High
Cost	< $1,000	Low
Dimensions of Laser Sheet Coverage	2ft x 6ft	High
Pothole Size Detection	Recognize potholes at least 4 inches in diameter	High
Laser System Weight	< 5 lbs	Medium

Table 2.4.2: Component Specifications

Component	Parameter	Specifications	Priority
IR Camera	Frame Rate	90 frames/second	High
IR Camera	FOV	100°	High
RGB Camera	Frame Rate	> 60 frames/second	High
RGB Camera	FOV	80°- 100°	High
RGB Camera	Resolution	1920 x 1080	Medium
Raspberry Pi	RAM	8 GB	High
Powell Lens 1	Fan Angle	≥90°	High
Powell Lens 2	Fan Angle	≥30°	High
Planoconvex Lens	Focal Length	18mm (3 x BCC FL)	Low
Biconcave Lens	Focal Length	-6mm(⅓ PCX FL)	Low

Table 2.4.3: Housing Unit Specifications

Parameters	Specification	Priority
Housing Material	Polycarbonate	High
Mounting Compatibility	GoPro Compatible attachment (Rear Mounted)	Medium
Camera Cutouts	2 front facing cutouts for IR and RGB cameras	High
Weather Protection Rating	IP65 for protection against water and dust	High
Thermal Management	Vent slots to dissipate heat	Medium
Operating Temperature	0℃ to 45℃	High

Table 2.4.4: Mounting System Specifications

Parameter	Specification	Priority
Material	6061-T6 Aluminum	High
Geometry (Cross Section)	Hollow rectangular tube	High
Length	Tescoping: 0.4m - 1m	Medium
Fastening Method	Clamp locking levers	High
Vehicle Attachment Method	5 Suction cups with vacuum locks	High
Vibration Tolerance	Designed to avoid resonance	High
Natural Frequency	>30 with damping	High
Environmental Protection	Resistance to water and mechanical shock	Medium

Prototype Illustration

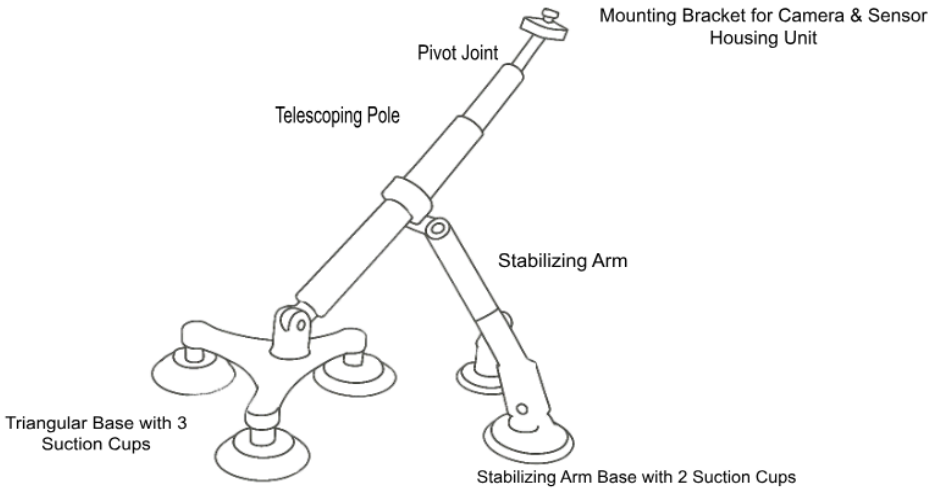


Figure 2.2: Mounting Device Prototype

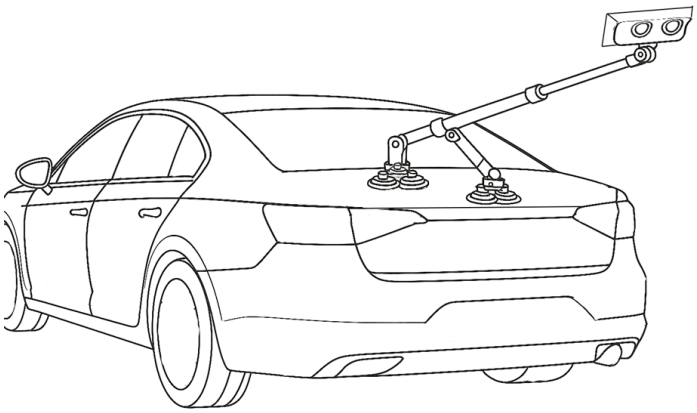


Figure 2.2.1: Mounting Device and Housing Unit Secured on Vehicle

The illustrated design for the mount system serves as a visual representation of the intended mounting assembly. The structure features a triangular base supported by three suction cups, each essential for providing a stable anchor to the vehicle's surface. The three suction cups in the rear deliver primary load support, while the two positioned at the base of the stabilizing arm

absorb torque and prevent tilt during sudden movements or turns. This secure mounting is critical to ensure accurate data collection while the vehicle is in motion. Extending from the base is a telescopic pole, designed to offer adjustable height and reach. The use of a modular, extendable pole and support arm ensures adaptability across different vehicle types and operating conditions.

The extendable pole allows for the mounted housing unit, containing the sensor and camera, to be positioned at an optimal vantage point. This ensures clear capturing of road imagery. The mounting bracket allows for quick mounting and includes a pivot joint, allowing for flexibility to determine the optimal and precise angle for surface scanning. Together, these components form a cohesive system that supports our project's goals: accurate pothole detection, reliable video capture and AI detection, and seamless integration to our mobile app. This prototype not only communicates mechanical function, but also reinforces the mount's role in ensuring accurate and high quality data acquisition while on the move.

2.5 Block Diagrams

2.5.1 Hardware Diagram

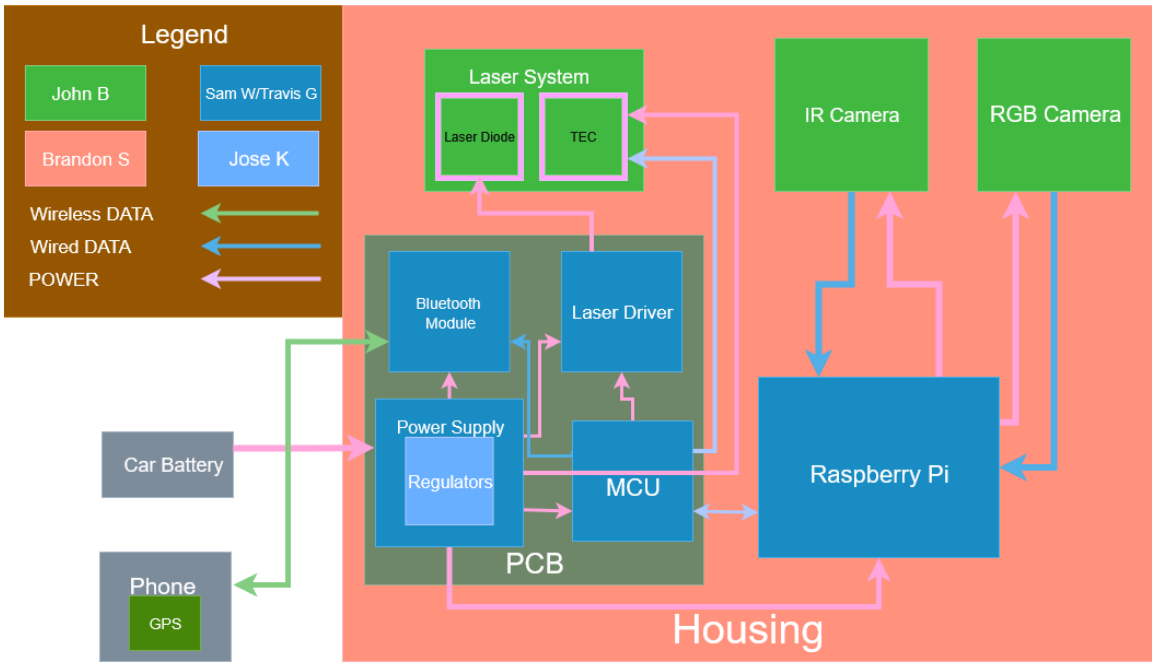


Figure 2.3: *Hardware Block Diagram*

The hardware block diagram outlines the system's hardware components and their interactions. Power from the car battery is regulated through a custom PCB that supplies the microcontroller, laser driver, and Raspberry Pi. The laser

system projects an IR sheet onto the road. The laser driver, controlled by the MCU, powers the laser diode. An IR camera detects discrepancies in the IR sheet and the RGB camera captures an image of the suspected pothole, sending data to the Raspberry Pi for image processing. Detected potholes are transmitted via Bluetooth to an iOS application with relevant data such as GPS coordinates and the time at which the pothole was detected.

2.5.2 Software Flowchart

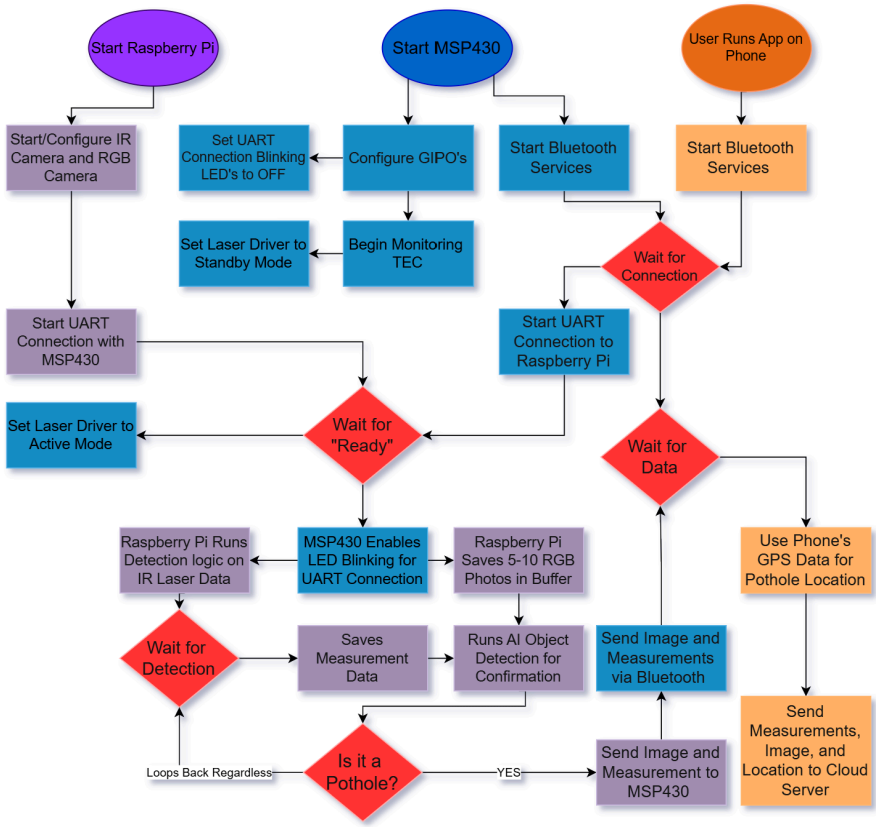


Figure 2.4: *Software Flowchart*

2.5.3 Flowchart Description

The software system begins with the MSP430 MCU and the Raspberry Pi starting up. The MSP430 is responsible for initializing hardware-level controls and Bluetooth communication, while the Raspberry Pi manages imaging and AI detection. After startup, the MSP430 configures GPIOs, sets the laser driver to standby mode, begins monitoring the TEC (thermoelectric cooler), ensures all blinking LEDs are set to off, and establishes a Bluetooth connection with the user's device. Once a Bluetooth device is paired, it waits for data to be transmitted and establishes a UART connection with the Raspberry Pi. Meanwhile, the Raspberry Pi initializes both the infrared camera and the RGB camera, and then establishes its UART connection with the MCU. Both devices wait for UART communication to be acknowledged. Once the UART connection

is established, the MSP430 enables a blinking LED to indicate UART activity and sets the laser driver to active mode.

The Raspberry Pi then begins running the laser line deviation detection logic using the infrared camera stream, while separately saving the most recent 5–10 RGB frames from the RGB camera into a buffer. This ensures that high-resolution visual context is available if a pothole is suspected. By only triggering AI image detection when the infrared laser system detects a potential pothole, the system significantly reduces computational load and avoids constant inference. This approach not only prevents performance bottlenecks but also allows the AI to process frames at higher speeds.

When the IR-based deviation algorithm detects what it thinks is a pothole, it calculates its diameter and triggers an interrupt to initiate AI-based image detection for confirmation. The program is multithreaded, allowing the IR detection and AI confirmation processes to run effectively at the same time. In a separate thread, the trained YOLOv5n model processes the buffered RGB images to determine whether the detection was in fact a pothole. If the AI fails to confirm the detection, the data is discarded, and the system returns to monitoring. If the AI confirms a pothole, the Raspberry Pi sends the measurement data and the RGB image over UART to the MSP430, where it's then sent over Bluetooth to the user's device. The app then uses the phone's GPS to tag the location, and all data is uploaded to a cloud server.

2.5.4 Mobile Application Software Block Diagram

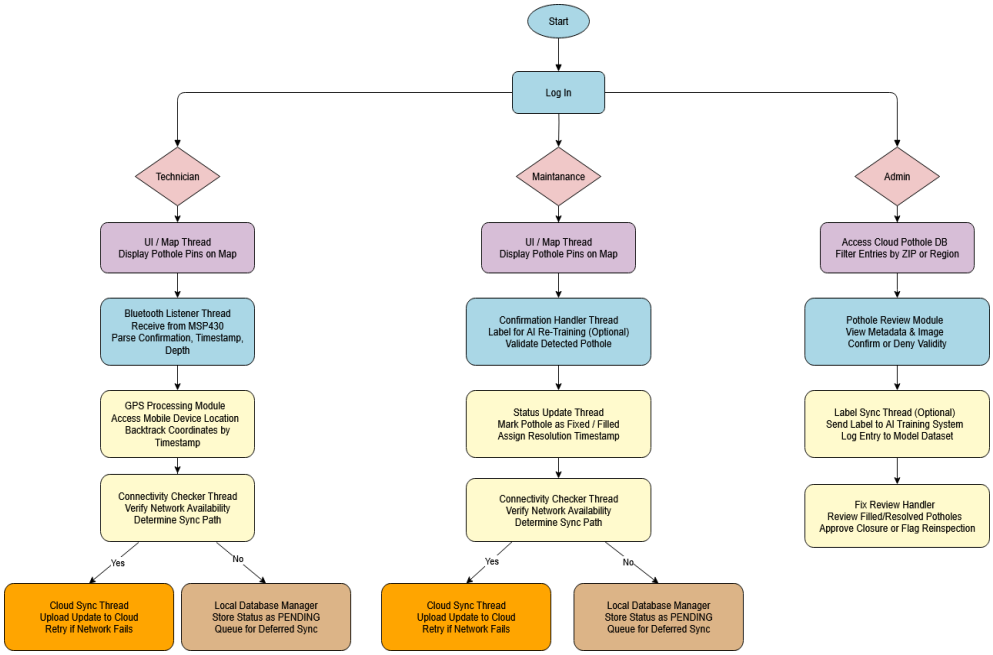


Figure 2.5: *Mobile Application Block Diagram*

2.5.5 Application Flowchart Description

The mobile application system begins operation once the user launches the app and Bluetooth services are initialized. The application immediately begins scanning for available MSP430-based devices using the standard Bluetooth protocol stack. Once a compatible device is detected, the system establishes a secure connection and transitions to data reception mode.

Upon successful pairing, the Bluetooth Listener Thread activates and continuously monitors the communication channel for incoming data. The MSP430 sends detection flags, measurement data (such as pothole depth), and a timestamp corresponding to when the detection occurred. The thread parses these payloads and forwards the extracted data to the GPS Processing Module.

The GPS Processing Module is responsible for associating geographic coordinates with the pothole event. Since the MSP430 lacks onboard GPS, the application uses the mobile device's GPS sensor to determine its location. If a pothole timestamp is received while the device is in motion, the module performs a backtracking procedure to identify the location that matches the original timestamp. This ensures that even if there's a short delay in transmission or a momentary GPS dropout, accurate tagging can still occur.

Next, the Cloud Uplink Handler evaluates internet connectivity and determines the appropriate data path. If connectivity is available, the Cloud Sync Thread is triggered, which packages the timestamp, location, depth, image, and user ID into a structured payload for cloud storage. If connectivity is unavailable or the upload fails, the data is handed off to the Local Database Manager, where it is temporarily stored and flagged as PENDING. This local queue is continuously monitored so that once a connection is restored, unsynchronized entries can be pushed to the cloud without user intervention.

The application maintains a multithreaded architecture to manage these operations independently and concurrently. In addition to Bluetooth and GPS functions, the UI / Map Thread updates an interactive map interface with pothole pins representing both confirmed and recently reported hazards. Each pin is associated with metadata, allowing for visual confirmation, live tracking, and administrative filtering.

In parallel, the Administrator Thread provides a privileged interface for authorized users to manage pothole reports across a broader geographic scope. Upon accessing the application, an administrator can filter reports by region or ZIP code, allowing them to focus on specific areas with a high concentration of unresolved entries. The Pothole Review Module is then used to inspect the submitted metadata, including images, location, and timestamps. Administrators can confirm or deny detections, which in turn improves downstream model training and quality control.

Confirmed entries may be queued for machine learning by the Label Sync Thread, which logs review outcomes for future retraining of the AI detection model (typically handled offline due to hardware limitations). Administrators also have access to the Fix Review Handler, which allows them to audit potholes marked as "fixed" or "filled" by field technicians. This component ensures accountability by offering one final checkpoint before a pothole is removed from the active system.

This modular and thread-based design allows the mobile application to serve multiple user types—drivers, maintenance personnel, and administrators—while maintaining data integrity and responsiveness even in constrained or offline conditions.

2.5.6 Mobile Application Design Diagram

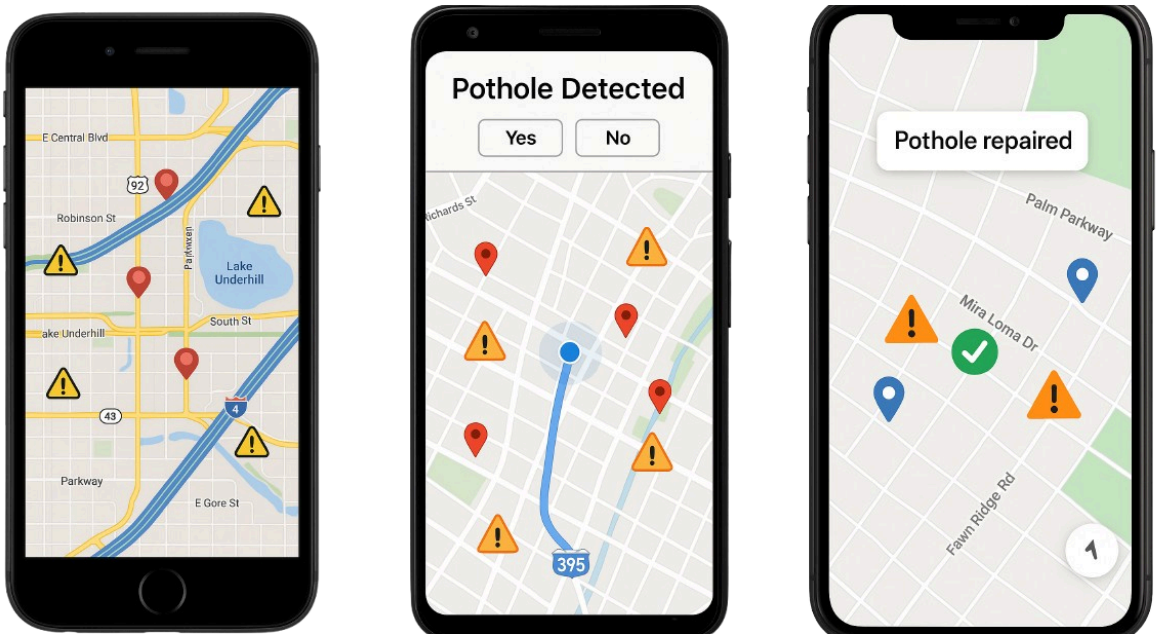


Figure 2.6: *Mobile application design*

The Mobile Application Design Diagram illustrates a conceptual design of the mobile application interface. The application uses a map-based visualization to display pothole locations with intuitive, color-coded markers. Red pins indicate confirmed potholes, yellow warning icons represent suspected or unreviewed detections, and blue pins denote the user's current location. This layout enables users and maintenance personnel to quickly assess road hazard density in real time and plan navigation or inspection routes accordingly.

In the center screenshot, the application prompts the user for manual confirmation when a new pothole is detected, allowing for optional AI validation input without compromising safety or requiring constant interaction. The rightmost screenshot demonstrates how resolved potholes are updated within the

UI, using green check marks to indicate successful repair. This feedback loop not only informs the user of system responsiveness but also assists administrators in verifying maintenance progress across the region. The interface is designed for simplicity and clarity, supporting quick decision-making in both driver and technician modes.

2.6 House of Quality

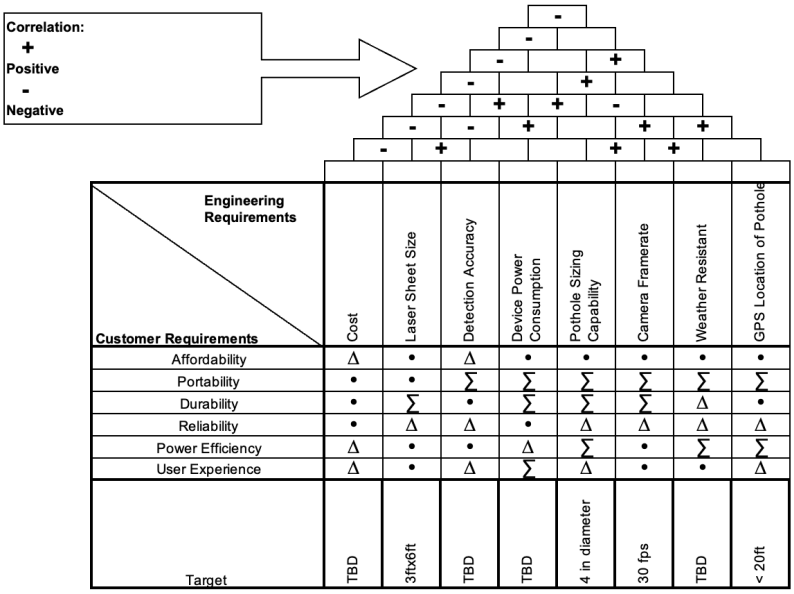


Figure 2.6.1: *House of Quality*

3. Research

3.1 Electrical Hardware Technology Comparisons

3.1.1 Microcontroller vs FPGA vs SBC

For this device, the design group is tasked with choosing between platforms that will be responsible for managing the other hardware components. These three technologies are the microcontroller (MCU), field programmable gate array (FPGA), and the single board computer (SBC). Each has their own set of benefits as well as their own set of setbacks.

There are many benefits to using an FPGA, this is because FPGAs are programmed using a hardware descriptive language, and they are programmed to perform one specific task. They implement true parallelism, which can run tasks simultaneously. They have a better processing power than the MCU and have a higher scalability.

Of course, there are also many setbacks when it comes to using an FPGA instead of a microcontroller. One of these setbacks is the complexity of programming the FPGA, there is a steep learning curve for using hardware descriptive languages, which would just add more time it would take to program the device.

The benefits of using a microcontroller is to manage low-level hardware tasks that do not require a lot of processing power. The responsibility of the microcontroller is to manage the laser driver, establish a UART connection with the Raspberry Pi, toggling GPIO status LEDS, monitoring the TEC, and sending GPS information to the bluetooth module. The microcontroller is there to ensure real-time responsiveness at the hardware level.

The microcontroller takes on a major role in the design because it is not only required to be a part of the design, but also because it allows the Raspberry Pi to handle the image processing while it makes sure certain peripherals have an established connection/power being supplied to them.

Microcontrollers usually have low-power characteristics, are reliable, and are good at interrupt handling. Which makes them an ideal choice for what the device is trying to accomplish. Having a microcontroller handle the control logic and enabling consistent coordination in the system is essential.

Table 3.1.1*: MCU vs FPGA*

Criteria	MCU	FPGA	SBC
UART Communication	Excellent	Poor	Good
Programming Difficulty	Excellent	Fair	Fair
Power Efficiency	Excellent	Poor	Poor
Real-Time Control	Good	Excellent	Poor
Cost	Excellent	Poor	Fair
Processing Power	Good	Excellent	Excellent
Overall	Excellent	Fair	Good

In conclusion, the design group ultimately decided to go with utilizing an MCU to do the logistical work of the design. And the group also decided to use an SBC to handle the AI image processing.

Microcontroller (MCU)

MSP430FR2355

The design team's first option when it came to choosing an MCU was the MSP430FR2355 or MSP430 as it's commonly known. This MCU is a low-power, 16-bit microcontroller developed by Texas Instruments. This MCU features an MSP430 CPU core operating at up to 24 MHz, as well as 32 kB of non-volatile FRAM and 4 kB of SRAM for quick memory access. The device supports multiple communication interfaces such as UART, I2C, and SPI, which is beneficial for the design team, because the team may have to change the way data is sent and received, offering a lot of flexibility. This device features 44 I/O, a 12-bit ADC, timers, and advanced power-saving modes. This device also offers an on-chip 32 kHz RC oscillator as well as an external 32 kHz crystal oscillator. [28]

ESP32 WROOM 32

The ESP32 WROOM 32 is another option that the design team also considered when it came to choosing an MCU for the device. This device features both bluetooth and Wi-Fi capabilities, however, due to design constraints, the design team is unable to utilize these features. The ESP WROOM 32 also features a D0WD-V3 chip, which features a dual-core 32 bit Xtensa LX6 processor with clock speeds up to 240 MHz. This device integrates 4 MB of external SPI flash memory and includes multiple serial interfaces as well such as UART, SPI, and I2C. The minimum power supply voltage of this device is 3V with a maximum power supply voltage of 3.6V, and a minimum typical current draw of 2.8 mA and a maximum typical current draw of 379 mA at its peak.

MSP430FR6989

The last MCU option the design group considered for our design was the MSP430FR6989 made by Texas Instruments. This MSP430 model is a 16-bit low-power microcontroller that features a CPU core that runs up to 16 MHz, with 128 KB of non-volatile FRAM and 2 KB of SRAM which means fast and low-energy memory access. This device includes a 12-bit ADC with up to 16 channels, multiple serial interfaces, including multiple UART, SPI, and I2C ports. This device supports up to 83 GPIOs, and includes an integrated LCD driver capable of handling up to 320 segments, in case the project design calls for an LCD display. The features of this MCU make it a flexible and effective option, coupled with the fact that the design team has experience with this device, it makes it a very viable option for the project.

Table 3.4.1: *MCU Comparison*

Criteria	MSP430FR2355	ESP32 WROOM	MSP430FR6989
Core Architecture	16-bit RISC	32-bit Xtensa dual-core	16-bit RISC
Power Supply Voltage	1.8V-3.6V	3.0V-3.6V	1.8V-3.6V
GPIO	44 pins	26 pins	83 pins
FRAM	32 KB	4 MB Flash	128 KB
SRAM	4 KB	520 KB (internal RAM)	2 KB
Current Draw	~100-350 uA/MHz	~20-130 mA	~100-300 uA/MHz
Operating Temperature	-40 - +85 ℃	-40 - +85 ℃	-40 - +85 ℃
Overall	Excellent	Good	Good

SBC Comparison

Raspberry Pi 5 vs BeagleBone AI-64

An integral part of this system is image processing using trained AI models that can confirm whether a discrepancy on the road is in fact a pothole. This makes choosing a device for handling image processing very important. There are many options when it comes to choosing such a device, but two devices ultimately stood out, the Raspberry Pi 5 and the BeagleBone AI-64.

Raspberry Pi 5

The Raspberry Pi 5 is powered by a 64-bit quad-core Arm Cortex-A76 processor operating at 2.4 GHz, offering a 2-3x increase in CPU performance over the previous Raspberry Pi 4. This performance gain is complimented by an enhanced 800 MHz VideoCore VII GPU, dual 4Kp60 HDMI outputs, and an updated Image Signal Processor (ISP) that provides advanced support for high-resolution cameras. These upgrades significantly improve the system's capabilities for desktop workloads and extend its applicability to industrial and embedded use cases.

A key advancement in the Raspberry Pi 5 architecture is the introduction of the in-house designed RP1 “southbridge” chip, which delivers a substantial increase in I/O performance and functionality.

Aggregate USB bandwidth has more than doubled, resulting in faster data transfers to external storage and high-speed peripherals. The previous two-lane 1 Gbps MIPI camera and display interfaces have been replaced with dual four-lane 1.5 Gbps MIPI transceivers, tripling the total bandwidth and supporting any combination of up to two cameras or displays.

BeagleBone AI-64

The BeagleBone AI-64 is built around Texas Instruments’ TDA4VM system-on-chip (SoC) from the J721E family, part of the K3 Multicore SoC architecture designed for automotive, industrial, and robotics applications. At its core, the SoC features a dual-core 64-bit Arm Cortex-A72 processor that can run up to 2.0 GHz, delivering up to 24,000 DMIPS of performance.

The BeagleBone AI-64 also features a variety of specialized processing units, like the C71x DSP with 80 GFLOPS, multiple C66x DSP cores, and a Matrix Multiplication Accelerator (MMA) capable of delivering up to 8 TOPS (Tera Operations Per Second) for advanced AI interfacing, machine vision, and signal processing with a low-power envelope, making this platform adaptable to edge-computing applications. The SoC integrates a set of hardware accelerators and peripherals to support industrial and automotive use cases. A Vision Processing Accelerator (VPAC) and a Depth and Motion Processing Accelerator (DMPAC) enable real-time processing of high-resolution camera data for applications such as advanced driver-assistance systems (ADAS) and robotics.

The integrated GPU, multi-standard HD video encoder/decoder, and flexible display interfaces (eDP, DSI, DPI) support complex multimedia workflows. The SoC also provides dual four-lane MIPI CSI interfaces for camera inputs, which can handle simultaneous multi-camera operations. The TDA4VM also emphasizes functional safety and security. Safety features include dual lockstep Cortex-R5F microcontrollers, error-correcting code (ECC) protection on critical memory, watchdog timers, and temperature monitoring sensors. An effective security architecture supports secure boot, trusted execution environments (TEE), and hardware cryptographic acceleration (AES, SHA, RSA). These features help ensure system integrity and reliability, even in harsh environments.

When it comes to connectivity, the platform includes multiple high-speed interfaces such as PCIe Gen3, USB 3.0, and Gigabit Ethernet, along with I/O like CAN-FD, SPI, I2C, UART, and GPIO. Simplified power management with on-die LDOs and adaptive voltage scaling (AVS) helps reduce the complexity and cost of the system. All of these features make the BeagleBone AI-64 a powerful platform for real-time vision analytics, deep learning applications, and embedded control.

Table 3.4.4: *Development Board Comparison*

Criteria	Raspberry Pi 5	BeagleBone AI-64
Main CPU	Quad-core 64-bit ARM Cortex-A76	Dual Cortex-A72
Frequency	2.4 GHz	2.0 GHz
AI/Vision Accelerators	VideoCore VII GPU at 800 MHz	C71x DSP, C66x DSPs, and MMa up to 8 TOPS
Interfaces	PCIe 2.0 x1, USB 3.0 x2, GbE, microSD	PCIe Gen3 x4, USB 3.0, GbE, dual CAN, PRU ICSS
Memory and Storage	LPDDR4X up to 16 GB	4 GB DDR4 + 16 GB eMMC
Power Consumption	Up to 6-7 W peak	Not Specified
Cost	~$80 (8 GB) - $132 (16 GB)	~$228
Overall	Excellent	Good

In conclusion, the design team has chosen the Raspberry Pi 5 as the SBC for this project. This is due to its comparable specs with the BeagleBone AI-64, but the key difference is the cost. The Raspberry Pi is much cheaper, which in this project is a major specification.

3.1.3 Device Power Source

At first, our group considered using a detachable and chargeable battery pack that would be a part of our device which would increase our device's portability feature. This would also create a safer testing environment, because not all users are comfortable wiring a device to a fuse tap in their vehicle. Power stability could also improve, as the input power would likely be less affected by vehicle voltage noise.

There are also some cons powering the device via battery. Having to remember to remove it and charge it approximately every day is not convenient, and is just another thing for our user to do each day. Runtime would be much more limited due to the fact that the battery would die relatively fast to wiring to a fuse tap, depending on the size. For the purposes of this device, and how it

would be used on a daily basis, the pros of using a battery weren't really priorities for the project.

Another option we had was utilizing the battery from the car, creating a DC connection via fuse tap, and using DC-DC buck converters to step down the 12V to the appropriate levels for the device's peripherals.

There are many benefits to using the car's battery as a power source. One of these benefits is the runtime relative to using a removable battery, which would require frequent recharging. The power supplied by the car's battery should be more than enough to power our device and its peripherals. The overall system requires less space due to not having a battery pack attached/inside the system.

There are also some cons with using a direct connection to the car's battery as a power source. The device would be less portable, as mentioned above, not every user is comfortable wiring a device to a fuse tap in their vehicle. The power is also less stable and predictable compared to that of using a battery to power the device, meaning the PCB is likely to be more complicated because the device is not using a completely steady source of power. Both are completely viable options for the device, each posing many pros and cons as well. For the purposes of what this device is hoping to achieve, and the ideal way that this device is to be used. The design group chose to use a DC-DC connection, utilizing the vehicle's battery.

Table 3.1.3: *Energy Source*

Criteria	Car Battery	Battery Pack
Runtime	Excellent	Fair
Cost	Excellent	Fair
Size and Weight	Excellent	Poor
Power Quality	Fair	Excellent
Portability	Poor	Good
PCB Design Complexity	Fair	Good
Overall	Good	Fair

In conclusion, our group decided to utilize the car's battery as the power source for our device. We chose this because of the convenience that the user would experience versus using a battery pack, which would die much quicker.

Switching Regulators vs Linear Regulators

A crucial piece of technology that our design will utilize are regulators. The two types of voltage regulation technology that the design team narrowed their choices down to are switching and linear regulators. Both voltage regulating technologies provide a stable voltage output, but take different approaches in doing so.

Switching regulators use an electronic switch that rapidly toggles on and off, transferring energy rapidly through inductors and capacitors in order to control the output voltage. Since switching regulators are constantly switching between on and off, a minimal amount of energy is dissipated as heat, this means the efficiency is quite good relative to linear regulators.

On the other hand, switching regulators require inductors and capacitors, which makes the total voltage regulation design more complex. This also makes the design physically larger, leading to less space on the PCB in the design team's case.

Switching Voltage Regulators

LM2576S-5.0

The first 5V voltage regulator the design team considered was the LM2576-5.0, a buck voltage regulator made by Texas Instruments, which was recommended by an advisor. This is a common voltage regulator that has been widely used for many years due to its reliability. This voltage regulator operates at a relatively low switching frequency of about 52kHz, it is known to be fairly easy to implement, and is known for its robustness.

This voltage regulator's efficiency and physical footprint are less favorable than the other voltage regulators that were considered for this device. Its low switching frequency requires larger external inductors and capacitors, which results in a bulkier design. This voltage regulator is known to generate more heat than other, new generation voltage regulators, which poses a threat to surrounding components that are sensitive to heat.

Since this voltage regulator is known to be dependable, even at the cost of efficiency, the design team will still consider it when it comes time for the final design. And since the design will not be in a small, enclosed space, the heat that is generated from this voltage regulator should not pose such a vital threat to the system or its components.

LMR33630

The second voltage regulator the design team considered was the LMR33630. This is also a buck voltage regulator created by Texas Instruments, but the key difference between this voltage regulator and the LMS2576-5.0, is that this voltage regulator allows the design team to adjust the output voltage according to whatever voltage is required by the peripherals or sensors. This is a significantly improved design when it comes to flexibility of output voltage when compared to the previous voltage regulator

For the purposes of this design, the fact that this regulator can accept inputs from 3.8V to 36V is a huge pro due to the range of a typical car battery. This means this voltage regulator can handle both low voltage and higher voltage events that can be caused by a car's battery.

This voltage regulator provides a continuous output current of 3A, which is enough amperage to supply our peripherals with sufficient current. This voltage regulator also operates at a much higher switching frequency than the LMS2576-5.0 at 400 kHz, giving faster transient responses to load changes, and also resulting in smaller inductors and capacitors, making the cost and physical footprint less than the previously mentioned voltage regulator.

The LMR33630 uses advanced high-speed circuitry that allows it to step down an input voltage as high as 20V to provide a stable 3.3V output, while maintaining a switching frequency of 2.1 MHz. It can regulate a 3.3V output voltage when it is getting an input voltage as low as 3.8V.

This regulator automatically switches between PWM (Pulse Width Modulation) and PFM (Pulse Frequency Modulation) depending on the load. When it experiences a heavier load, it uses the PWM mode with a constant switching frequency for stable performance. When it experiences lighter loads, it uses the PFM mode with diode emulation and discontinuous conduction (DCM), which minimizes input current and maximizes efficiency.

This voltage regulator also includes internal loop compensation, which simplifies the design by reducing external components and design time compared to regulators that require external compensation.

Packaged in an ultra-compact VQFN with wettable flanks, the LMR33630 minimizes parasitic inductance and resistance. This enables higher efficiency, reduces switch node ringing, and significantly lowers electromagnetic interference (EMI). Its symmetrical VIN and PGND pin layout further helps cancel input current magnetic fields, contributing to reduced EMI generation.

MP2338

The last 5V regulator the design team looked at is the MP2338 from Monolithic Power Systems. This voltage regulator is a 3A synchronous buck regulator with a high switching frequency of 1.4 MHz which improves transient response or a faster reaction to changes in load, and enables the use of small inductors and capacitors, which would make it better for keeping the size of the PCB down.

This voltage regulator can handle an input voltage range of 4.5V to 24V, which makes it a very real contender for this design, because of the volatility of the supply voltage received from the car's battery. One feature of this voltage regulator is internal soft-start, which gradually ramps up output voltage when enabled. This is important because it prevents large inrush currents at startup, it also reduces stress on input capacitors which can help prevent brownouts.

Another feature of this voltage regulator is undervoltage lockout, which means this regulator will not turn on if the supply voltage drops below a certain threshold. This can help prevent erratic behavior in conditions where the supply voltage is dropping, it can also help the system because if the system is not getting enough supply voltage, this can lead to logical errors. Another feature of the regulator is cycle-by-cycle current limiting, which can help protect the regulator and load by limiting the current on each switching cycle. This is important because it can help prevent damage if the system is receiving too much load.

Table 3.4.3: *Voltage Regulator Comparison*

Criteria	LM2576S-5.0	LMR33630	MP2338
Max Output Current	3A	3A	3A
Input Voltage Range	7V-40V	3.8V-36V	4.5V-24V
Output Voltage	Fixed (5.0V version)	Adjustable (.8V-Vin)	Adjustable (.925V-Vin)
Switching Frequency	52 kHz	400 kHz (typical), up to 2.1 MHz	1.4 MHz
Efficiency (Peak)	65%-75%	90%-95%	90%-95%
Light Load Efficiency	Poor (no PFM)	Excellent (PFM + DCM mode)	Good (PFM mode)
Synchronous	No, external	Yes, integrated	Yes, integrated

Design	Schottky Diode required	MOSFETS	MOSFETS
Protection Features	Basic OVP, thermal shutdown	OVP, UVLO, current limit, thermal	UVLO, current limit, thermal
Soft-Start	No	Yes	Yes
Thermal Performance	Heatsink required at higher loads	Minimal heat	Good, needs airflow at 3A
Cost	$3.46	$2.09	$1.55
Overall	Poor	Excellent	Fair

In conclusion, as you can see from the table above, the LMR33630 beats the other two regulators by many factors. Thus, the design team has chosen the LMR33630 as the regulator for this device. This is due to its flexibility and adjustability of its output voltage.

3.1.2 Pothole Data Access

The initial design of the device utilized the bluetooth module attached to the Raspberry Pi. Due to design constraints, the design group had to consider other options of data transfer/storage.

The bluetooth module would be soldered to the PCB, and have a corresponding LED in order to be sure that it is receiving power. The bluetooth module will receive data from the MCU, where it will then be in contact with the user's phone in order to mark a GPS location via an app.

One option also considered was local data storage, such as storing pothole data including GPS location on an SD card or another physical memory. Issues with this are both the extra time and the extra effort required to retrieve such data. The user would have to upload the data onto a server or app at the end of each day, which would not be very efficient.

Some pros of using an SD or other physical memories would be the cost, power, and simpler implementation of the memory storage itself. It would also eliminate the need for constant wireless connectivity if the user was in an area with poor reception, or if the user's phone was dead. It would be a viable option for a backup in case the bluetooth option had failed.

Table 3.1.2: *Pothole Data Access Comparison (Incomplete)*

Criteria	Bluetooth Communication	Local Data Storage	Cellular Communication
Range	Excellent	Poor	
Bandwidth	Good	Excellent	
Real-Time	Excellent	Poor	
Power Use	Excellent	Excellent	
User-Friendly	Excellent	Poor	

As mentioned above, initially, using the pre-installed bluetooth module on the Raspberry Pi was heavily considered, but this was not possible due to design constraints. After further thought and instruction, the design group felt that using a bluetooth module separate from the Raspberry Pi, was the best option.

Bluetooth Module

Microchip Technology RN4871-I/RM140

Upon initial research into finding a bluetooth module for the project, the design team came across the Microchip Technology RN4871-I. This is a compact BLE (bluetooth low energy) module that integrates a bluetooth 5.0 baseband controller and RF amplifier, but is essentially limited to BLE 4.2 features due to its firmware stack.

This device draws about 10.0 mA during transmission at 0 dBm at a temperature of around 25 degrees Celsius, and peaks at about 13 mA under higher temperatures of about 75-85 degrees Celsius, this device draws the same amount of current as these when receiving as well. This device has a supply voltage of about 1.8 to 3.6 V as well, which is typical for a bluetooth module of this size. [41]

RayTac MDBT50Q-1MV2

The second bluetooth module that the design team found was RayTac's MDBT50Q. The 1MV2 just indicates that the module uses an antenna located on the chip rather than the PCB, which is the desired model for this design. The MDBT50Q is a bluetooth 5.2 stack module which features bluetooth low energy or BLE. This design is based on the Nordic nRF52840 SoC solution, which

incorporates GPIO, SPI, UART, I2C, I2S, PMD, PWM, ADC, NFC, and USB interfaces for connecting peripherals.

This device features a 2.4GHz, bluetooth 5, IEEE 802.15.4 transceiver. It also supports data rates of 125 kbps, 500 kbps, 1 Mbps, and 2 Mbps, with 1 and 2 Mbps using the proprietary 2.4 GHz frequency. This device also features a flash memory of 1 MB, and a RAM of 256 KB, along with an ARM Cortex – M4 processor with FPU, 64 MHz.

The power management system of this device is also flexible, with a supply voltage range from 1.7V to 5.5V, on-chip DC-DC regulators and LDO regulators, as well as the ability to power other peripheral devices with a voltage of 1.8V to 3.3V, making this device very versatile. [40]

u-blox NINA-B306-01B

The final bluetooth module that the design team considered for the device is the u-blox NINA-B306-01B. This bluetooth module has comparable specifications to the previously mentioned bluetooth modules above, more specifically to the RayTac bluetooth module. This is because this bluetooth module is also based on the Nordic nRF52840 solution.

This bluetooth module also has a clock speed of 2.4 GHz, a flash memory of 1 MB, and a RAM of 256 KB. The supply voltage range for this device ranges from 1.7 to 3.6 V, with a current draw in TX mode from 4.9 to 14.1 mA, depending on the power output used, which ranges from 0 dBm to 8 dBm. The current draw for RX mode is about 4.8 mA.

The features of this bluetooth module make it incredibly attractive for the design of the system. However, this device's lack of flexibility when compared to the RayTac bluetooth module makes it difficult to choose.

Table 3.4.2: *Bluetooth Module Comparison*

Criteria	**Microchip Technology RN4871-I/RM140**	**RayTac MDBT50Q-1MV2**	**u-blox NINA-B306-01B**
Clock Speed	2.4 GHz	2.4 GHz	2.4 GHz
Flash/RAM	N/A	1 MB/ 256 KB	1 MB/ 256 KB
Input Voltage Range	1.8-3.6V	1.7-5.5V	1.7-3.6V

Current Draw (TX/RX Mode Peak Current)	13 mA/ 13 mA	2.7-32.7 mA/ 3.7-11.1 mA	4.9-14.1 mA/ 4.8 mA
Cost	$9.24	$6.15	$8.13
Temperature Range	-40 - +85 ℃	-40 - +85 ℃	-40 - +85 ℃

3.2 Optical Technology Comparisons

3.2.1 Detection Technology

LiDAR

LiDAR makes use of the travel time of a laser pulse to determine the distance to an object. This is done by illuminating an area, usually with a pulsed laser, and measuring the time it takes for each pulse of light to return to the system.. LiDAR often takes advantage of extremely quick pulsed lasers and avalanche photodiodes to create a 3D image of the target area. Due to the nature of light being extremely quick LiDAR systems need to use photodiodes that can take many measurements within a short period of time such as an avalanche photodiode. The components needed to use LiDAR require extreme precision and very fast response times and are oftentimes very expensive.

LiDAR systems primarily use near-infrared light as there is far less potential harm to humans if a stray beam were to enter their eyes. As LiDAR is often used in urban environments the use of visible light also introduces the potential of creating dangerous distractions. Additionally, LiDAR devices have very large FOVs allowing for a lot of area to be scanned even when low to the ground

LiDAR is very effective at high speeds since light is extremely fast and many points of data can be collected very quickly with negligible distance between data points. This means that the collected data will be very detailed and detections would be easier to make. However, the large amount of data collected using LiDAR presents an issue when it comes to processing power and data storage as it will be more taxing on a computer to receive data points at such a high speed.

Since LiDAR mapping is entirely based on time of flight information there is little risk that it will make a false detection of something like an oil slick since the light being received by the system will have extremely negligible differences in time of flight.

Machine Vision

Machine vision involves the use of a camera and software to identify information about a desired area. There are many things that can be measured with the use of machine vision but for this project it would be used to measure whether or not something is in the area being scanned and how large that something is. The camera receives the light reflected from the road and converts that information into digital data. This data is then transferred to a computer to be processed by a trained algorithm.

The camera converts the image data into digital data. The machine vision program converts the image data into a bitmap. Using this bitmap data the program can look for inconsistencies in the received light to detect objects or depressions on the road. However, when using visible light machine vision requires a lot of processing power. This is because the image data contains 3 different sets of values for red, green, and blue light. By converting the images into grayscale the image data is significantly reduced in size. Unfortunately, since the data is only in black and white the program is unable to differentiate between a pothole and any other road depression. To combat this, another camera can be used to capture images whenever a detection is made and a computer vision program can be used to confirm whether or not the depression is a pothole. The addition of another camera that only activates upon detection allows for better object detection without needing a lot of processing power.

At high speeds it can be difficult to measure the target area with a high resolution as when using a camera with a low fps there may be a large distance between frames when traveling at average road speeds. However, increasing the fps will require an increased amount of processing power which may be difficult to achieve without a dedicated CPU.

The illumination system allows the program to gather more data from the grayscale images by increasing the contrast of the scanned area. There are a variety of ways to illuminate the target area including lasers, or LEDs. To properly illuminate the area there are a variety of things to consider such as angle, wavelength of light, and distance from the scanned area.

The camera will need a proper lens system to capture the transmitted light. When picking the lens system there are a lot of important characteristics to take into account. In order to capture a meaningful portion of the lane the camera will need to have a large FOV so the use of a wide angle lens may be beneficial, however this would introduce distortion to the image so there needs to be a balance. The most common types of camera sensors are Charge-Coupled Devices (CCD) and Complimentary Metal Oxide Semiconductors (CMOS). Both of these sensors work primarily in the visible spectrum of light but can still detect Near-Infrared light. This allows a lot of flexibility when it comes to choosing the operating wavelength of the system.

Detection System Comparison

For this project there are a variety of factors that we need to take into account when designing the detection system. One of these factors is the image resolution when the device is operating at high speeds. With machine vision the resolution is reliant on the frame rate of the camera being used, however with an increased frame rate there is the need for more processing power. LiDAR is entirely based on time of flight for a laser beam meaning the distance between each data point being taken is reliant on how quickly the laser pulses unless using a continuous wave laser which would make the distance between data points negligible. For these reasons LiDAR beats out machine vision when it comes to resolution.

Since our device will be relatively small we will be severely limited on processing power, as such it is important that our detection system requires very little. The processing power required for machine vision is quite variable due to many different parameters of the device but it is very possible to keep the processing power low. LiDAR on the other hand requires, on average, a lot of processing power. This is due to the large amount of data that is collected and processed by the LiDAR system. Most LiDAR systems require a dedicated CPU which is just not something that is feasible for this project. For these reasons machine vision beats out LiDAR in terms of processing power.

Another important factor to consider is the FOV of the device. For this project we will need to scan a large portion of the road 6-10ft and we will be limited to being relatively close to the ground. Most LiDAR devices operate at high FOVs without distortion. Cameras on the other hand will start to have distortion at high FOVs. Due to the lack of distortion LiDAR wins out over machine vision when it comes to FOV.

The ability to operate at a variety of wavelengths is an important factor for this project as having the option to change wavelengths would offer a safety net on the off chance that one design falls flat. LiDAR usually operates in the NIR region and the wavelength cannot be changed easily as it would likely come packaged in one device. Machine vision allows for a variety of wavelengths to be used with very minimal changes needed for the device to work at other wavelengths. As such machine vision beats out LiDAR in terms of its flexibility.

Since we are working on a tight budget the issue of cost is the most important one. For machine vision we will need to acquire a camera with an appropriate lens that can work at a high fps along with a light source and the necessary equipment needed to operate it. For LiDAR we would need to acquire and build an array of avalanche photodiodes, and a laser system to illuminate the road. Due to the high cost of the equipment needed for LiDAR, machine vision would be ideal when considering the budget.

Table 3.2.1: *LiDAR vs Machine Vision*

Criteria	LiDAR	Machine Vision
Resolution	Excellent	Good
Processing Power	Poor	Good-Excellent
FOV/Distortion	Excellent	Good
Flexibility	Fair	Good
Cost	Fair	Good
Overall	Fair	Good-Excellent

While it may struggle when it comes to distortion and high speed resolution, machine vision is a much better option for this project as it can perform on nearly the same level as LiDAR whilst being cheaper, more flexible and requiring significantly less processing power. The increased flexibility of machine vision also allows for quick changes if something doesn't work out.

3.2.2 Wavelength of Light

Visible Light

Visible (VIS) light is the portion of the electromagnetic spectrum that is visible to the average human being. It is generally defined as the wavelengths of light from about 400nm to about 700nm. Visible light is used in a variety of detection and/or scanning systems as well as countless other devices. Because of its visibility it is extremely easy to work with and is often used for testing before switching to a non-visible wavelength of light. Visible light comprises a little over 40% of the light radiated by the sun and due to the small size of the visible spectrum it is the most densely radiated by the sun.

Infrared Light

Infrared (IR) light is the portion of the electromagnetic spectrum that ranges from around 780nm-1mm. Infrared light makes up a majority of the light emitted by the sun at around 50% but due to how large the IR spectrum is it is less densely radiated by the sun than the visible wavelengths. The infrared spectrum can be broken down into 5 main categories: Near-infrared (NIR), Short-wavelength infrared (SWIR), Mid-infrared (MIR), Long-wavelength infrared (LWIR), and Far-infrared (FIR).

NIR is the smallest portion of the infrared spectrum, ranging from about 780nm-1400nm, and is able to be captured by most visible cameras. It is most commonly used in fiber optic telecommunications and night vision devices. The

SWIR spectrum ranges from about 1.4μm-3μm and is most commonly used for long range telecommunications. The MIR spectrum ranges from about 3μm-8μm and is often used for heat seeking applications. LWIR ranges from about 8μm-15μm and is used primarily for thermal imaging as these are about the wavelengths of light emitted by objects around -80-89°C. Thermal imaging devices don't need any kind of illumination since the light is emitted with heat. FIR light ranges from about 15μm-1mm and is mostly used for medical purposes such as skin treatments.

Wavelength Comparison

As SWIR, MIR, and FIR are not very practical for imaging devices they will not be considered for the comparison. We will be comparing the VIS, NIR, and LWIR spectrums for use in our illumination system. There are 5 factors that this comparison will be based on: cost, noise, ease-of-use, simplicity, safety.

Due to the constraints of our project, keeping the cost low is very important. NIR and visible light are very commonly used making them much cheaper than LWIR devices, however they require some sort of illumination. Although LWIR devices remove the need for illumination they are still more expensive than VIS and NIR. Between VIS and NIR illumination sources VIS devices tend to be cheaper meaning that visible light wins in terms of cost.

In order to make accurate detections we need to be able to clearly make out the differences in light levels on the road and as such noise is a big concern for our project. With visible light there is a lot of noise from the sun because it is the most densely radiated spectrum of light by the sun. NIR also suffers from a lot of noise from the sun as it is still very densely radiated by the sun but there is less noise than for the visible spectrum. LWIR has the most amount of "noise" as almost everything being imaged will be releasing light in this spectrum, however this is how LWIR imaging is done without illumination. Due to the sheer amount of LWIR light it can be hard to make accurate measurements which means NIR wins out in terms of noise.

Ease-of-use is very important for testing our design. Given that visible light is easy to see it is also very easy to test with and make fine adjustments. The same cannot be said for NIR or LWIR as they both require special instruments to be able to see. NIR is slightly easier to work with than LWIR as it can be viewed with most visible cameras. Visible light is by far the easiest to work with and wins in terms of ease-of-use.

Keeping the project simple will help to make mistakes more manageable and changes easier to make. Visible and NIR light are the same in terms of simplicity, they both require some sort of illumination source and a camera to view the light. LWIR is very simple with it only requiring a camera thus making it the most simple wavelength to work with.

The last but potentially most important factor to consider is the danger posed by using any of these wavelengths in a public environment. NIR and LWIR are both invisible to the human eye which makes them highly unlikely to distract drivers on the road. Visible light can cause a distraction for other drivers but it also has the advantage of being noticeable and reactable for the human eye. LWIR is the safest as no light is being sent to the road and the risk of distracting or injuring a bystander is not present. NIR is the next safest option as it does not run the risk of distracting a bystander but it has an increased chance of permanent eye damage by a stray beam. However, considering that by the time the light hits the ground it will be extremely weak there is a miniscule chance of the visible or NIR light causing damage to the bystander's eye. For these reasons I believe that NIR and LWIR are the safer options.

Table 3.2.2: *Wavelength Comparison*

Criteria	Visible Light	Near-infrared Light	Long-wavelength Infrared Light
Cost	Good	Good	Poor
Noise	Fair	Good	Poor
Ease-of-use	Excellent	Good	Fair
Simplicity	Fair	Fair	Excellent
Safety	Good	Excellent	Excellent
Overall	Good	Excellent	Poor

While it might be a bit difficult to work with and slightly more expensive than visible light, NIR light is safer to use on the road and there is less noise to worry about, making it an excellent choice for this project.

3.4.2 Optical Component Selection

Camera Selection For Machine Vision Detection System

Camera Requirements

When selecting the cameras for this project there are a variety of requirements to consider. Since this device will be operating at high speeds our camera will need to be able to capture good images at high intervals with minimal image blur. Additionally, our project requires that we scan a large area of the road from a set distance so the camera we pick will need to have a high FOV lens to capture the desired area. Due to the large amount of processing power needed for machine vision it would be ideal for the camera to record the image data in

monochrome to save on bandwidth and processing power. Having a high resolution is important for our design but due to constraints on the processing power of our device a camera that has variable resolution with a high limit is preferred. For this project we will be running cables through a car's physical firewall so a camera that can transfer data through without the need to convert to a different cable type would be preferred. Finally, in the detection system, we will be using infrared light so our camera will need high sensitivity to infrared while still being relatively cheap.

To capture video at high speeds our camera will need to have a high framerate. Our target max speed for operation is 40mph which equivalates to just under 60 feet per second. Since our laser sheet will be about 2 feet in length we will need our camera to have a framerate of at least 30fps. It is possible that we will want to increase our accuracy so a camera with the option to increase frame rate would be preferred. When working at high speeds image blur starts to become a concern. Image blur occurs when image data for individual pixels is recorded at different times, this is dependent on whether the camera uses rolling shutter or global shutter. When a camera sensor uses rolling shutter technology an image is generated by reading the pixel data row by row, while working at high speeds the image data from the starting pixels and the later pixels will be recorded at different positions causing the resulting image to blur. Sensors with global shutter technology read the image data of every single pixel at the same time resulting in a blurless image. A camera with a sensor that has global shutter is much preferred for this project.

For the camera to capture the entire desired area of the road it will need a lens with a high field of view(FOV). Based on our calculations, the camera lens must have a vertical field of view (FOV) of at least 28°, and a horizontal FOV of at least 74° or 103°, depending on whether the target area is 2×6 ft or 2×10 ft, respectively, when viewed from a height of 4 ft above the ground. If the lens that comes with the camera does not fit these specifications then a camera with an easily swappable lens will be acceptable.

Machine vision requires a sizable amount of processing power meaning that the ideal camera can make up for this in some way. To keep the necessary processing power low we may need to limit the resolution of the camera but having a low resolution will decrease the accuracy of our measurements so a camera with variable resolution and a high max resolution is ideal. Having the option to change resolution will make quick changes to the design easy to implement. Another way to limit the amount of data coming from the camera is to find a camera that records in monochrome. Without the color data the amount of information being transferred is reduced and less processing power is needed to process the image data.

The image data from the camera needs to be sent over the length of a car in cables small enough to fit through a car’s physical firewall. To avoid signal loss our camera should be able to transfer data across the car without the need to

convert from one cable to another. We have decided that USB 2.0 and 3.0 are the best choice for our device as they have minimal loss and can be very cheap even at long lengths. As such it is preferred that the camera we choose has a built in USB connection or at the very least can be converted to USB if need be.

The illumination system for the device will be using near-infrared light which can be captured by visible light cameras although most filter out infrared light. To make infrared light detection easier our camera should not have an IR filter if it is a visible camera. Additionally, if a replacement lens is required it should not have an IR filter so as to inhibit the functioning of the camera. Another option is to use a near-infrared camera sensor with a custom made mount for the sensor and lens.

Camera Comparison

When choosing a camera for the detection system there were 3 main candidates: the Raspberry Pi Camera Module 3 NoIR Wide, the Arducam B0332, and the Mira050 NIR sensor with custom mount and lens setup. All of these options fit the basic needs of our project but none of them are perfect. The pros and cons of each camera will be evaluated in the table below.

The Raspberry Pi Camera Module 3 NoIR Wide is designed to be used with a Raspberry Pi making it easy to implement in our design. It consists of an IMX708 sensor as well as a high FOV lens already mounted to the device. The IMX708 sensor captures images in color and uses rolling shutter, making it suboptimal for the high speed environment of our device. The lens has a horizontal FOV of 102° and a vertical FOV of 67°, on an M12 mount, making it suitable for the smaller target area and potentially the larger target area since the difference is minimal. This camera sends and receives information via a CSI ribbon cable however it can be converted to USB with an adapter. This camera can be operated at or above 30fps in 480p, 720p, and 1080p resolution. There is no IR cut filter in this camera so it can pick up infrared light quite easily.

The Arducam B0332 is also designed to work on the Pi and other similar devices and as such is easy to implement in the design. It consists of an OV9281 sensor and a 70° FOV lens. The OV9281 image sensor captures monochrome images and uses global shutter, making it a good option for high speed applications. The lens has a horizontal FOV of 70° and a vertical FOV of about 48° which wouldn't work for either target area however, the camera has an M12 lens mount making swapping lenses quite easy. The Arducam has a built-in USB connection for data transfer which is ideal for our design. This camera operates at up to 100fps in 240p, 360p, 800x600, and 1280x800 resolutions and has no built-in IR filter.

The Mira050 image sensor is designed to capture near-infrared light and is the best option in terms of detecting NIR light. It captures images in monochrome and uses global shutter. This sensor can operate at up to 120fps

with a max resolution of 600x800. The biggest downside to this option is that it requires that we build a custom mount for the sensor and lens, we would also need to create a board that allows the data to be transferred through a USB cable. This is the most complicated option but it meets the detection requirements the best.

Table 3.4.5: *Camera Comparison*

Criteria	Raspberry Pi NoIR Wide	Arducam B0332	Mira050 NIR Image Sensor
Framerate	50 @ max res	100 All	120 All
Resolution	1080p; Variable	800p; Variable	800x600; Variable
Shutter Type	Rolling	Global	Global
Lens Field of View	102° (H), 67° (V); M12 Mount	70° (H), 48° (V); M12 Mount	None
Image Color	Color	Monochrome	Monochrome
Cable Connection	CSI Ribbon Cable	USB 2.0	None
IR Sensitivity	Good	Good	Excellent
Cost	Good (USB Adapter)	Good (Alternative Lens)	Poor (Many Additional Costs)
Overall	Good	Excellent	Fair

While its resolution is a bit low and it needs a replacement lens, the Arducam B0332 is the clear winner here. All of the options have good IR sensitivity, acceptable frame rates, and resolutions but the Arducam stands out as it uses global shutter and a built-in USB connection. The Mira050 image sensor is best suited for NIR imaging but it requires a ton of extra work and additional costs that other options do not. The Pi camera is an excellent option but the use of rolling shutter is too big of an issue for this project.

Filter Selection

Type of Optical Filter

Since our project utilizes infrared light and we are using a visible light camera it is very important that the visible light is filtered out before reaching the camera. It is important that the filter only lets through light of the same wavelength as our laser in order to minimize noise. The filter also needs to work in a high FOV optical system like that of our device.

We considered 2 different types of optical filters for this design: a dielectric bandpass filter, and an absorptive color bandpass filter. These both have their advantages but only one fits our project, the table below is used to explain what we picked.

Dielectric bandpass filters are composed of many thin films of dielectric material that make use of constructive and destructive interference to only allow a specific wavelength of light through. Because of this they have very little error often in the 10s of nanometers. However, at high angles of incidence(AOI) the center wavelength of the passband decreases due to the increased optical path length changing the constructive interference condition. Due to this, these bandpass filters severely limit the FOV of an optical system. These filters tend to be quite expensive.

Absorptive color filters rely on a material's properties to absorb light of different wavelengths than the desired wavelength resulting in the desired light passing through and undesirable light being absorbed by the filter. These filters tend to have very broad passbands since absorption properties are quite broad. The advantage to absorptive color filters is that their properties are minimally affected by the angle of incidence and they do not tend to limit the FOV of an optical system. Absorptive color filters tend to be much cheaper than dielectric filters.

Table 3.4.6: *Optical Filter Comparison*

Criteria	Dielectric Filter	Absorptive Color Filter
Passband Size	Excellent	Fair
High AOI Operation	Poor	Excellent
Cost	Fair	Excellent
Overall	Fair	Excellent

While they do have a broad passband, absorptive color filters are a better choice for our design overall as they do not limit the FOV of our device as much as a dielectric filter would, they are also a bit cheaper which will help us stay on budget.

Absorptive Color Filter Selection

The process of selecting the absorptive color filter was entirely based on what was available and met the requirements for our design. In order to be chosen the filter will need to center on a wavelength of light in the NIR region with a reasonably small passband. The 2 options I landed on were the Hoya

RT830 Colored Glass Bandpass Filter and the Schott RG905 Colored Glass Bandpass Filter.

The Hoya filter is centered on a wavelength of 830nm with a full width at half maximum(FWHM) of 260nm and does not pick up light with a wavelength smaller than 700nm. The Schott filter on the other hand is centered on a wavelength of 880nm with a much larger FWHM, it also picks up some light around the 500nm range.

Table 3.4.7: *Absorptive Filter Comparison*

Criteria	Hoya RT830	Schott RG905
Passband	Good	Poor
Center Wavelength	Excellent	Excellent
Cost	Good	Fair
Overall	Excellent	Fair

The Hoya bandpass filter is a much better option for our design as it has a much smaller passband and costs less than the Schott filter.

3.2.3 Illumination Source

Light Emitting Diode (LED)

Light Emitting Diodes consist of 3 regions: an n-type, negatively charged, semiconductor, a p-type, positively charged, semiconductor, and a depletion region between them. By applying a current through the LED from the p-type semiconductor to the n-type semiconductor the depletion region begins to narrow as the negatively charged electrons and positively charged holes move closer together. With this movement the electrons and holes start to come into contact with each other and release energy as photons in a process called spontaneous emission.

Due to the variable nature of spontaneous emission the light emitted by the LED has a high amount of variability. LEDs emit light omnidirectionally, are relatively low cost and low energy. In order to generate enough light many LEDs can be used in an array, this will also decrease the coherency of the emitted light as there will be multiple light sources in the array.

Laser

Laser is an acronym meaning “Light Amplification by Stimulated Emission of Radiation.” As is evident by the name, lasers take advantage of stimulated

emission to generate light. In order to achieve stimulated emission a laser requires 3 major parts: a pump source, a gain medium, and a laser cavity.

The gain medium is the material used to achieve a population inversion. They are designed to have energy levels separated by the energy of photons of the desired wavelength of light. By exciting the gain medium with the pump source the higher energy level becomes populated with excited electrons until there are more electrons in the excited state than the ground state, this is known as population inversion. However, when population inversion is achieved most of the excited electrons will spontaneously emit photons. This is where the laser cavity comes in, the gain medium sits inside of the laser cavity with 2 mirrors on either end; these mirrors have different reflectivities with one mirror being 100% reflective and the other being less than 100% reflectance.

As the light emits from the gain medium it bounces off of the mirrors in the laser cavity and back into the gain medium, when the light returns to the excited electrons it causes them to emit photons of the same wavelength. This leads to a cascading effect where the light coming through the medium increases and with it the amount of excited electrons being hit causing more and more photons to be emitted. As the amount of light in the cavity increases eventually the light is able to escape the cavity through the mirror with <100% reflectivity. Because of the way that the photons are emitted there is very little variance in the wavelength of the emitted light. This results in the laser emitting coherent light of a single wavelength which.

Illumination Technology Comparison

There are three important variables to consider when picking an illumination source for this project: cost, coherency, and divergence. Keeping the cost low is important to this project as we are funding the project ourselves. LEDs are a very cheap option even in the infrared light ranges. Lasers are generally quite expensive relative to LEDs but they have gotten a lot more affordable for the average person, although IR lasers tend to be a bit more expensive. LEDs beat out lasers in terms of the cost.

In order to be able to properly image the light with minimal interference we will need to have a light source that is very coherent. While LEDs don't have a lot of variance when it comes to the wavelength of light emitted an array will be needed which introduces a lot of incoherency. Lasers however produce a single wavelength of light with minimal variance and as such are very coherent. Lasers are much more coherent than an LED array.

Having an illumination source with very little divergence is important for this project as being able to control the light into the specific area we desire is important to save on the necessary power for the device. LEDs emit light omnidirectionally and thus have a lot of divergence which will result in a lot of the emitted light not being picked up by our detection system. Lasers however have

very little divergence and the light is very easy to manipulate allowing us to prevent wasting energy on light that we are not imaging. For these reasons the laser wins out over LEDs in terms of divergence.

Table 3.2.3: *Illumination Source Comparison*

Criteria	Light Emitting Diode	Laser
Cost	Excellent	Fair
Coherency	Fair	Excellent
Divergence	Poor	Excellent
Overall	Fair	Excellent

While it may cost a bit more the laser light is coherent with very little divergence making it a much better option overall.

3.3 Software Technology Comparisons

3.3.1 Computer Vision Models

Single-Stage vs Two-Stage Object Detection Models

To determine the best object detection model for our design, it is important to consider the specific constraints and needs of the system. Typically, the main factors when selecting a model include speed, accuracy, hardware demand, model complexity, and training effort. In our case, accuracy and hardware demand are the top priorities. Accuracy is critical because the model will directly influence whether sensor data is kept or discarded, which has a significant impact on the overall reliability of the system. Hardware demand is equally important, as the detection model will run on a Raspberry Pi, which has more limited processing power compared to what many object detection models are built for [17]. Since our design does not require real-time detection, speed is still relevant, but not the most important factor. The system cannot experience major slowdowns, but the software is not dependent on detections being faster than the frame rate of the RGB camera.

Object detection models are generally grouped into two categories: single-stage and two-stage models [11],[13]. Two-stage models operate in two parts: a region proposal stage and an object classification stage. During the region proposal stage, the model uses a lightweight algorithm, such as selective search, to scan through the image and identify areas that might contain objects. At this point, the model does not attempt to identify the objects; it only highlights regions that stand out, such as a person or a tree in a field. The second stage then refines these proposed bounding boxes to fit more tightly around the objects

and tries to classify them. Common examples of two-stage models include R-CNN, which uses selective search, and Faster R-CNN, which replaces selective search with a Region Proposal Network (RPN) [12]. Two-stage models generally offer higher accuracy, better object localization, and improved noise filtering [13]. However, they tend to run slower, place more demand on hardware, and involve a more complex training process due to the need to train both stages separately [13].

Single-stage models, in contrast, perform both object localization and classification in a single pass [11]. These models take the full image and divide it into a grid, with each grid cell responsible for detecting objects whose centers fall within it. A backbone Convolutional Neural Network (CNN) is used to convert the image into a feature map, which helps the model pick out useful information [13]. Each cell predicts one or more bounding boxes and assigns coordinates and dimensions to them. The model then calculates a confidence score for each bounding box, representing the likelihood that an object is inside it. In addition, it produces a probability for each class type, showing how likely the object belongs to each class. Some models also use anchor boxes to account for different object sizes and shapes [16]. Any bounding boxes with low confidence scores are filtered out, and finally, Non-Maximum Suppression (NMS) is applied to remove overlapping boxes in favor of the one with the highest score [16]. Popular single-stage models include YOLO (You Only Look Once), which typically uses Darknet as its backbone, and SSD (Single Shot Detector), which often uses VGG-16 [14],[15],[16]. The benefits of using a single-stage model include faster processing, easier training, less demand on hardware, and better handling of various object sizes [14],[19]. The main drawbacks are slightly lower accuracy overall and a reduced ability to detect partially hidden objects [13], [18].

Table 3.3.1: *Single-Stage vs Two-Stage Object Detection Models*

Criteria	Single-Stage	Two-Stage
Speed	Good	Poor
Accuracy	Fair-Good	Good-Excellent
Complexity	Fair-Good	Poor
Hardware Demand	Fair-Good	Poor
Flexibility	Good	Excellent
Training Time	Fair	Poor
Overall	Good	Fair

Although accuracy is our main concern, the 10-15% improvement offered by two-stage models doesn't make up for the trade-offs in hardware demand, speed, complexity, and training time. Two-stage models place too much strain on a 64-bit quad-core Arm Cortex-A76 unless major hardware upgrades are made [12],[13],[17],[21]. While speed isn't a top priority, since this isn't a real-time detection system, a two-stage model can still take over 10 seconds to process just 5-10 images [12],[19]. This delay could seriously impact the accuracy of recording pothole locations, as we rely on the user's GPS data immediately after detection. Based on the data, it's clear that a single-stage model is a much better fit for our design.

YOLOv5 vs SSD Single-Stage Object Detection Models

To determine the most suitable single-stage detection model, we focused on two of the most widely used options: YOLO (You Only Look Once) and SSD (Single Shot Multibox Detector). Our goal was to find a model that offers high accuracy, low hardware demand, reasonable speed, and manageable training complexity. Since our task is binary (determining whether a specific object is present in the frame or not), this simplicity may positively affect model performance compared to more complex, multi-class detection tasks [25].

The SSD300 architecture begins its detection process with a backbone network, such as VGG-16, which extracts feature maps from the input image [27]. These maps are then passed through a series of extra convolutional layers that progressively reduce in spatial resolution. Higher-resolution layers are better at detecting smaller objects, while lower-resolution layers focus on larger ones [27]. Default anchor boxes are applied at each feature map location, and for each anchor, the model predicts bounding box offsets and class probabilities. Finally, Non-Maximum Suppression (NMS) is used to remove low-confidence and overlapping detections. SSD300's advantages include lower hardware requirements, a simpler and more compact design, fast initialization, and low memory usage [16],[25],[27]. However, it also has some drawbacks: it struggles with small object detection, generally has lower accuracy, offers limited built-in training features, and is increasingly considered outdated when compared to more modern models like YOLOv5.

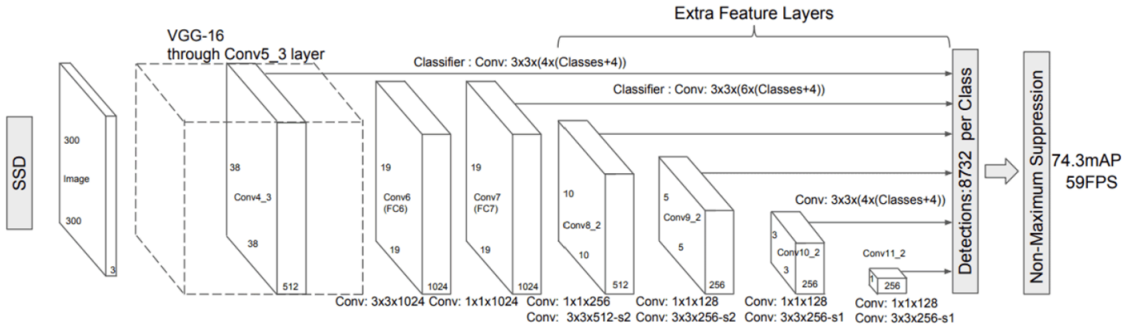


Figure 3.1: Architecture of the SSD300 model, utilizing VGG-16 as the backbone. Adapted with permission from [19].

The YOLOv5 architecture is divided into three main parts: the Backbone, Neck, and Head. The model begins with the Backbone, typically CSPDarknet, which is responsible for extracting important features from the input image. These features are then passed to the Neck, where information is combined from different layers using a Path Aggregation Network (PANet) and Spatial Pyramid Pooling-Fast (SPPF). This helps the model detect objects of varying sizes more effectively. The Head is responsible for predicting bounding boxes, confidence scores, and object classes. After this, anchor boxes are applied, and Non-Maximum Suppression is used to remove overlapping and low-confidence detections. The advantages of the YOLOv5 architecture include support for multi-scale training, automatic anchor tuning, higher accuracy, faster inference, and an active development community [23],[24],[26]. However, its disadvantages include greater architectural complexity and slightly higher hardware requirements compared to simpler models such as SSD300 [23],[26].

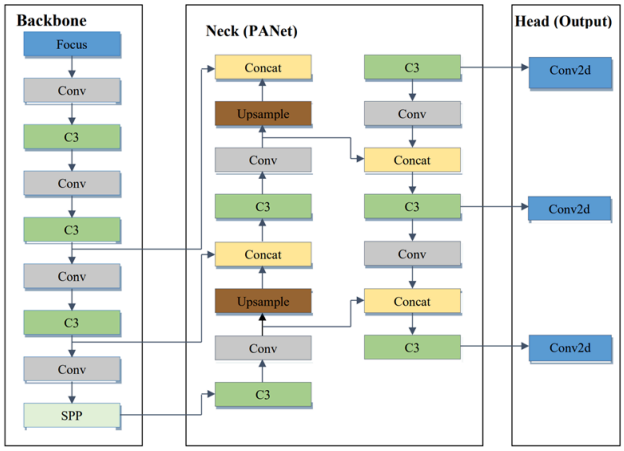


Figure 3.2: *YOLOv5 Architecture (Nepal & Eslamiat, 2022)[14]. Adapted with permission from [15].*

Table 3.3.2: *YOLOv5 vs SSD300 Object Detection Models*

Criteria	YOLOv5	SSD300
Accuracy	Good–Excellent	Fair
Small Object Detection	Good	Poor
Hardware Demand	Fair-Good	Excellent
Speed	Good	Good

Training	Excellent	Poor
Ease of Implementation	Fair	Good
Multi-Scale Detection Support	Excellent	Fair
Community & Documentation	Excellent	Fair
Complexity	Fair	Good
Overall	Good	Fair

The difference in hardware demand is not a strong enough reason to choose the SSD300 model. While the YOLOv5 model does require slightly more hardware resources, its significant improvements in accuracy, small object detection, training support, and multi-scale detection make it a far better choice for our application [17], [21],[27]. YOLOv5 comes in five versions (n, s, m, l, and x), ranging from smallest to largest in size and complexity [18]. The YOLOv5n model is capable of running at approximately 5 frames per second on the Raspberry Pi 5, while still allowing the two cameras and the laser detection algorithm to operate in parallel [17],[23]. On the other hand, YOLOv5s can achieve around 2 frames per second and offers up to a 20% increase in accuracy. However, the added resource demand makes it nearly impossible to run any other detection logic on the Raspberry Pi at the same time. While YOLOv5s remains a viable option, YOLOv5n offers a better balance of speed, efficiency, and system-wide performance, making it the most suitable model for our design [17],[23].

YOLOv5n Optimization Methods for Raspberry Pi

The native deep learning framework used for the YOLOv5 model is PyTorch, which is primarily intended for building, training, and testing neural networks. While PyTorch is easy to use and requires no additional setup, it presents significant limitations when it comes to runtime efficiency on devices like the Raspberry Pi 5. Specifically, PyTorch lacks optimizations for embedded hardware, which leads to lower inference speeds and inefficient use of system resources [21],[23]. To improve performance without noticeable accuracy loss, several common optimization methods can be applied. These include converting the PyTorch model to a proprietary inference engine like ONNX Runtime or NCNN and applying INT8 quantization. It's worth noting that while using a USB hardware accelerator (like a Coral TPU) is another viable approach, the added cost of around $100 makes it less attractive for budget-sensitive designs [26].

Exporting the PyTorch model to ONNX has been shown to increase inference speed by approximately 3 to 4 times on devices like the Raspberry Pi 4, with less than 1% accuracy loss in most cases [24],[25],[42]. ONNX enables

multithreading, quantization support, more efficient CPU usage, and reduced memory consumption, making it a well-balanced option in terms of performance and resource usage [23],[42]. The primary downside is the reduced flexibility, as the model must be exported and integrated into a slightly more complex pipeline.

For even greater optimization, the ONNX model can be converted to NCNN, a high-performance neural network inference framework designed specifically for ARM devices. NCNN delivers the fastest performance on CPU-only systems, making it ideal for final production deployment on mobile devices and Raspberry Pi boards [23],[24],[26]. However, NCNN can be complex to implement, offers less flexibility, and has limited community support compared to PyTorch or ONNX [23],[24].

Another major optimization strategy is quantizing the model from FP32 to INT8. This technique can be applied to both ONNX and NCNN formats. INT8 quantization reduces the numerical precision of weights and activations, significantly improving inference speed (up to 2–4× faster), cutting model size by ~75%, and lowering power consumption [22],[24],[42]. The tradeoffs include a slight drop in accuracy (typically <1%), a more complex workflow, and additional library or hardware dependencies depending on the runtime environment [23],[24],[42].

Table 3.3.3: *Object Detection Model Optimization Methods*

Criteria	PyTorch	ONNX	NCNN	PyTorch (INT8)	ONNX (INT8)	NCNN (INT8)
FPS	~1.5–2.5	~4-5	~7-10	~3-5	~8-10	~12-15
Accuracy Change	Baseline	Baseline	Baseline	~≤1% drop	~≤1–2% drop	~≤1% drop
Ease of Setup	Excellent	Good	Fair	Good	Fair	Poor
Memory Use	Poor	Fair-Good	Fair	Fair-Good	Excellent	Good - Excellent
Multicore	Poor	Good	Excellent	Fair	Good	Excellent
Overall	Poor-Fair	Good	Good	Fair-Good	Excellent	Excellent

Although it may seem that the best option is to use NCNN with INT8 quantization, the additional speed gain may not be worth the increased complexity involved in training and implementing this model. Instead, using ONNX with INT8 quantization should be sufficient to provide the memory

efficiency, speed, and multicore support needed for our application. With these optimizations, we expect to achieve less than 25% of the original memory usage, enough inference speed to run the model in under one second, and only a minimal, likely unnoticeable, drop in accuracy [16],[24],[42]. If our system eventually requires more performance, we can still transition to NCNN with INT8 quantization as a future upgrade path.

Dynamic vs Static INT8 Quantization of Object Detection Models

When applying 8-bit quantization to an object detection model, it's important to understand that there are two main methods for doing this. The first method is called dynamic INT8 quantization and the second method is called static INT8 quantization. Both methods convert the model's weights and activations from 32-bit floating point values to 8-bit integers [42]. The difference is how and when they do this. Choosing the best method depends on the developer's needs, and where they are in the design process.

With dynamic quantization, only the model's weights are converted to 8-bit integers ahead of time. Activities stay as 32-bit floating point values and their ranges are calculated at runtime. This makes setup much simpler, usually requiring just a single Python command to convert the model's weights to 8-bit integers. Since no calibration data is needed, this method is especially useful during the prototyping stage. Dynamic quantization typically boosts model speed by 2-3x, reduces RAM usage, and still maintains accuracy that's very close to the original FP32 version when run on edge devices [23],[42].

For static INT8 quantization, it converts both weights and activations to 8-bit integers before inference. This is done using something called calibration, which is the only way to quantize activations ahead of runtime. Calibration involves passing a set of sample data through the model to help it learn the typical range of activation values for each layer. These minimum and maximum values are then used to define factors that allow the model to run while only using integer operations. While this adds an extra step and increases complexity, it also gives the best performance. However, it does rely on using good calibration data, otherwise accuracy can decrease. Static quantization is best suited for final deployment when the model needs to be as fast and efficient as possible. It can improve inference speed by 2-4x, reduce model size and RAM usage by around 75%, lower latency, and greatly improve power efficiency [12],[23],[27],[42].

Table 3.3.4: *Dynamic vs Static Quantization*

Method	Speed	Latency	Accuracy	Setup	RAM	CPU	Overall
Dynamic IN8	2-3x faster	200-250 ms	Less than 1% drop	Good	Good	Fair	Good

Static INT8	2-4x faster	125-167 ms	Less than 1-2% drop	Fair	Excellent	Excellent	Excellent

While it may be more complicated to implement, static quantization is clearly the best option for models that need high speed, low RAM usage, and minimal CPU load [27],[42]. For ease of development, it's often better to start with dynamic quantization during the prototype stage, since the model is likely to change frequently. If the dynamically quantized model performs well and the Raspberry Pi can run it consistently without issues, then it may be perfectly acceptable to stick with dynamic quantization without needing further optimization [42].

3.3.2 Comparative Evaluation of Modern Mobile Development IDEs and Frameworks

As part of the pothole detection and tracking project , there is a mobile application that communicates, via Bluetooth, with the MCU responsible for controlling the system. The app allows users to review, and verify detections made by the system; on a user-friendly map . This requires a development environment that can use both Android and IOS mobile platforms, perform reliably, and allows for quick interface development. To find the best option, we reviewed several well-known development tools and frameworks.

The options reviewed include Android Studio, Xcode, Flutter, React Native, and .NET MAUI. Each of these tools offers several different features that affect development speed, compatibility, system requirements, and user interface consistency.

Android Studio

Android Studio is the main development environment for building Android apps using Kotlin or Java. It is supported by Google and gives a wide range of tools for debugging, interface design, and system profiling. Müller [35] found that Android Studio's Gradle build system is flexible and effective for managing complex Android applications.

Ivanović [33] analyzed several real-world Android applications and observed that performance bottlenecks and power consumption issues are often tied to common design patterns used in the Android Studio environment. While its integrated tools are comprehensive, the study suggests that code optimization within the Gradle structure is essential to avoid resource inefficiencies.

However, Android Studio can be demanding on computer resources. Developers using mid-range machines may experience slower performance, particularly when dealing with large projects. The Gradle build system, while capable, may slow down rapid development because of long compile times.

Android Studio is best used for projects focused entirely on Android rather than cross-platform solutions.

Xcode

Xcode serves as Apple's standard development environment for building applications tailored to their various platforms, including iOS, macOS, and watchOS. It provides comprehensive support for both Swift and Objective-C and comes equipped with a suite of essential tools for debugging, designing user interfaces, and testing. A study done by Tan [37], indicated that Xcode's integrated intelligent coding tools, such as code completion and navigation, significantly enhance developer efficiency, particularly when working with newer Swift projects. This comprehensive and integrated environment streamlines the development workflow, enabling developers to build and refine their applications effectively.

However, Xcode operates exclusively on macOS. This poses a challenge for development teams working across multiple operating systems. Managing larger applications within Xcode can be complex, partly due to the way it handles visual tools and interface files. Despite its limitations, Xcode remains the preferred development environment for Apple platforms due to its integration with Apple's native SDKs and the App Store deployment [37]. This allows developers to take advantage of the latest platform features and streamline the release process.

Flutter

Flutter is Google's UI toolkit designed to create applications for Android, iOS, web, and desktop using the Dart codebase. It is consistent for rendering and high performance across platforms, along with powerful tools for building visually rich and responsive user interfaces. A study by Lovrić[36], showed Flutter's reliable cross-platform performance and how features like hot reload significantly boost development speed. Hot reload works by injecting updated source code files into the running Dart Virtual Machine (VM).

A study by Lee found that Flutter outperformed Kotlin and React Native in areas like UI rendering and response time. [34] And while Abu Zahra and Zein noted that test automation frameworks work well with Flutter, they did observe some minor differences in test behavior across platforms[31].Some limitations present while using Flutter include that apps can be larger in file size, and the support for web and desktop platforms is still under development. Even so, Flutter remains well-suited for cross-platform development, especially when code reuse and interface consistency are high priorities.

React Native

React Native is a framework developed by Meta that allows mobile development using JavaScript and the React codebase. It connects JavaScript code to native components, which can help improve performance compared to web-based frameworks. In a study done by Biørn-Hansen [32] ,he observed that

React Native is useful for developers who already know JavaScript, allowing for faster and easier app developement

Uddin [38] performed a study of performance overhead in cross-platform tools and observed that React Native often introduces high -load delays due to its JavaScript-to-native bridge. Abu Zahra and Zein [31] also observed test inconsistencies between platforms when using automated test suites with React Native.

The main issues with React Native are related to performance in resource-intensive tasks and potential delays caused by its JavaScript-to-native bridge. Creating consistent behavior across platforms may also require writing platform-specific code, which can reduce some of the benefits of shared development.

.NET MAUI

.NET Multi-platform App UI (MAUI) is Microsoft's toolkit designated for building applications accross platforms including Android, iOS, Windows, and macOS. It provides support for the C# programming language and builds on the foundation of Xamarin.Forms. The framework is integrated into Visual Studio and is packaged as a suite of tools for designing user interfaces, debugging, and managing platform-specific configurations. A study by Abdullwahed and Tran [35] observed that .NET MAUI's unified project structure and integrated development tools help streamline both the creation and maintenance of applications across platforms, by simplifying cross-platform workflows within the .NET ecosystem.

One of .NET MAUI's key advantages is its project structure, which allows shared resources, styles, and logic to be managed more efficiently. Developers familiar with the .NET environment benefit from seamless access to .NET libraries, Azure services, and Blazor integration. However, as a relatively new framework, .NET MAUI is still under development. While .NET MAUI continues to be developed and offers a strong option for teams already working within the .NET ecosystem, projects that demand high performance or deep integration with device-specific features may still be better suited to native development tools.

Table 3.3.5 *Comparative Evaluation of Mobile Development IDEs*

IDE/ Framework	Cross Platform Support	Language(s) Used	Ease of Use	GPS Integration	Database Integration	Performance
Flutter	Yes	Dart	High	Easy via plugins	Easy (Firebase, SQLite, Hive)	High
Android Studio	No (Android only)	Kotlin/Java	Medium	Full native control	Flexible (Room, Firebase)	High
Xcode	No (iOS only)	Swift/Obj-C	Medium	Full native control	Comprehensive (Core Data, Firebase)	High

React Native	Yes	JavaScript	High	Moderate, may need bridging	Good (SQLite, Firebase)	Moderate
.NET MAUI	Yes	C#	Medium	Unified API abstraction	Good (SQLite, Azure, less Firebase support)	Moderate

Software Selection
Flutter

Based on the requirements of this project, Flutter was selected for mobile development. It allows deployment to both Android and iOS from a single codebase, reducing effort and improving consistency.

Flutter includes support for GPS and database features needed in this application. Location services and cloud storage can be added using well-supported plugins such as geolocator and cloud_firestore. These tools help implement real-time pothole logging and data synchronization without needing platform-specific changes.

The framework also includes tools for building user interfaces and handling background tasks. Its performance and available libraries match the scale of this project, making it a fitting choice for the mobile portion of the pothole detection system.

Each of the tools reviewed has strengths and weaknesses. Android Studio is powerful for Android-specific work but has performance drawbacks. Xcode is necessary for iOS development but is limited to Apple computers. React Native is fast and developer-friendly but may need additional platform-specific work. .NET MAUI supports C# and simplifies UI design, but it is still gaining maturity.

Flutter provides a practical and efficient solution for our project. It supports both Android and iOS from a shared codebase, offers useful tools for interface design, and performs well enough for the needs of our application. These reasons make it a good fit for building the mobile portion of our pothole detection system

3.3.3 Mobile Application Architectures: Analysis for Location and Data-Driven Applications

In evaluating options for implementing the mobile application of the project's system, it was also important to research all possible application architectures. Beyond the development tools themselves, we had to decide whether to deploy the application as a native app, web app, cross-platform app and hybrid app.

Each option offers different trade-offs in terms of device access, offline capabilities, performance, and application maintenance. Since the application needs to process GPS data, communicate with hardware through Bluetooth, and display real-time location on a map, it was essential to select a solution that supports responsive performance and consistent behavior across devices.

Web Apps (Including Progressive Web Apps)

Web applications particularly Progressive Web Apps (PWAs) are built to run within a web browser using technologies such as HTML5, JavaScript, and CSS. PWAs use special tools built into modern web browsers to do more than just show websites. These tools let the application work even when there's no internet, save information on the device, and use phone features like GPS to find your location. PWAs are being considered more consistently for cross-platform applications due to their simplified deployment, reduced overhead, and ability to run on multiple platforms from a single codebase [43].

Unlike native applications, PWAs are accessed directly through a URL, eliminating the need for distribution through app stores. This approach allows for a shorter development life cycle and easier maintenance. This also allows for lower overhead when frequent updates or wide accessibility are necessary. Their relatively small footprint reduces storage demands on the user's device. In a study by Lingolu and Dobbala [44], PWAs can enhance user engagement and simplify version management—qualities that are particularly valuable in civic and infrastructure-related applications where accessibility and update timeliness are essential.

Despite these benefits, PWAs still have many drawbacks when used in applications that require high-accuracy GPS, consistent hardware access, or low power consumption. Studies have shown that the real-time location accuracy of web apps can vary between browsers and devices, often falling short compared to native apps [44]. In addition, PWAs generally use more CPU and memory, which can lead to increased battery drain. In a study by Horn, he demonstrated that native applications consistently outperformed web apps in both power and system resource usage[45].

Although IndexedDB supports offline storage, it is limited by how much space browsers allow and doesn't support advanced features like spatial queries, which are often needed for complex government data tasks. Additionally, web apps are restricted from accessing more advanced device functions, including background location tracking and combining data from multiple onboard sensors, like the GPS, accelerometer, and gyroscope, to improve motion or location accuracy. These capabilities are essential for real-time data collection and infrastructure monitoring.

For Department of Transportation (DOT) efforts like pothole detection and infrastructure mapping, web apps may work well for administrative dashboards,

citizen reporting portals, or public-facing tools that need simple interactivity and easy access. However, their limitations in GPS precision, background data logging, and offline performance make them a poor fit for fieldwork. While web apps can complement native systems, they are generally not recommended as the primary platform for mission-critical tasks involving sensors and real-time data capture in transportation operations.

Native Apps

Native applications are developed for a specific operating system using languages tailored to that platform—such as Kotlin or Java for Android, and Swift or Objective-C for iOS. Once built, the app is compiled into code that runs directly on the device, giving it fast performance and full access to hardware features and built-in system functions. This results in better performance, more consistent behavior, and tighter integration with device capabilities.

In GPS-based government applications, native apps provide several important benefits. They deliver more accurate and reliable location data by using high-frequency GPS updates and platform-supported features that combine sensor data for better tracking. Horn [45] found that native apps consistently outperformed web apps in both power efficiency and system resource use. This makes a major difference for field workers who depend on their devices to last through long shifts without needing to recharge.

When it comes to storing data, native apps can use powerful offline databases like SQLite or SpatiaLite. These systems support spatial queries and allow the app to function fully even without internet access. Data can be synced automatically once the device reconnects, making them well-suited for remote areas or signal-dead zones. Native apps also support background tasks, push notifications, and built-in security tools like biometric login, which are useful in protecting sensitive data and keeping users informed.

From a transportation standpoint, native apps match the practical needs of Department of Transportation (DOT) operations. Field crews can use them on rugged devices to record pothole locations, take photos, and upload information directly to GIS databases. The high accuracy of native GPS tools ensures that location data is precise, important for scheduling maintenance, planning routes, and prioritizing repairs. Offline support adds an increased reliability by ensuring that data isn't lost when the network connection is weak or unavailable.

Despite these strengths, native development has its drawbacks. Creating and maintaining separate apps for Android and iOS takes more time, resources, and development skills. Once updated, the application must go through application store review processes, which, depending on the application store platform, can be a lengthy process—especially for a government application, as they have more stringent requirements. However, tools like Flutter now make it possible to write code once and deploy it as a native app on both platforms. This

helps balance the performance of native apps with the efficiency of cross-platform development.

In the end, native apps are the best choice for government projects that require high GPS accuracy, secure data handling, offline reliability, and full hardware access. Their features make them especially useful in tasks like pothole detection, road inspections, and infrastructure monitoring—where data quality and system dependability are key. While the development effort may be higher, the long-term benefits are often worth the investment, especially for field operations in the public sector.

Cross-Platform Apps

Cross-platform applications are built using tools like Flutter, React Native, and Xamarin. These tools allow developers to write most of the code once and use it on both Android and iOS. This shared approach helps save development time, and is useful for applications that need to be available on multiple platforms. These frameworks also give access to important device features like GPS, the camera, and local storage, making them more capable than basic web-based apps. Biørn-Hansen [40] notes that cross-platform development can significantly speed up deployment without causing major drops in performance.

Cross-platform frameworks create native like experiences by using built-in components from the operating system, which may lead to better performance and smoother user interfaces. This makes them a strong option for applications that need to collect data, track GPS locations, or store information offline.

Some limitations of cross-platform applications include the lack of support for advanced features.To implement these features, developers need to write custom native code. This can make the application more difficult to maintain as it becomes more complex. Performance of cross-platform applications may also suffer in applications with complex animations, graphics-heavy interfaces, or heavy background activity. Since cross-platform applications don't have access to the deep system integration and up-to-date protection features, their security is often as strong as other application implementation forms.

Overall, cross-platform frameworks offer a solid balance between cost, performance, and device access. They are a good fit for projects that need to work on both Android and iOS, especially when the goal is to keep development time and costs low without giving up essential mobile features.

Hybrid Apps

Hybrid apps are made using common web technologies like HTML, CSS, and JavaScript, and are placed inside a native wrapper using tools such as Apache Cordova or Ionic. This allows them to run on both Android and iOS devices while sharing a single codebase. Users can download them through app

stores just like regular mobile apps, but the experience often feels more like using a webpage inside an app shell.

These apps are typically used for projects that don't need deep interaction with the phone's hardware. While hybrid apps can connect to some native features like the camera, GPS, or file storage the level of access is usually limited. As a result, hybrid applications may not perform as well as native or cross-platform apps, especially when dealing with tasks that require real-time data processing or background activity. This can be a serious drawback for systems that depend on location tracking, sensor data, or offline logging, such as those used in transportation and public infrastructure reporting.

One of the main reasons teams choose hybrid development is cost. Since most of the code can be reused for both platforms, development is faster and more affordable, which is useful for smaller projects or simple applications. Some applications that display static data, company information, or collect basic form data can be built and deployed quickly without the overhead of full native development.

However, hybrid apps are not a good fit for more demanding tasks. They often struggle with performance, and updates to device operating systems may cause compatibility issues that require frequent plugin maintenance. This makes hybrid solutions less reliable for government or field-service applications where accurate data collection and uninterrupted functionality are critical. As Lingolu and Dobbala [44] explain, hybrid apps offer a budget-friendly path for mobile development, but they are better reserved for lightweight use cases that do not depend heavily on hardware integration or speed.

Table 3.3.6 *Core Mobile Application Types*

Application Type	Definition	Common Technologies	Key Use Cases
Native App	Built for a specific OS using platform-native languages (e.g., Swift, Kotlin).	Swift, Kotlin, Objective-C, Java	High-performance apps needing full device access
Cross-Platform App	Single codebase compiled into native binaries for multiple platforms.	Flutter (Dart), React Native (JS), .NET MAUI (C#)	Apps for both iOS and Android with shared business logic
Hybrid App	Web technologies wrapped in a native shell (e.g., using Cordova or Ionic).	HTML, CSS, JavaScript + Cordova/Ionic	Quick deployment apps with moderate device access
Web App	Runs in a browser, not installed, relies on internet and browser capabilities.	HTML, CSS, JavaScript	Public info portals, basic tools, mobile-friendly websites

Progressive Web App (PWA)	Web app enhanced with offline support, push notifications, and installability.	HTML5, JS, Service Workers, Web Manifests	Offline-capable public service tools and interactive sites

3.3.4 Comparative Analysis of Database Architectures for Government-Oriented Mobile Applications

MongoDB

MongoDB is a document-based NoSQL database developed to support flexibility and scalability in large-scale, data-diverse systems. It is available in two main editions: a free Community Edition and a feature-rich Enterprise Edition. MongoDB stores data in BSON format, which allows for semi-structured, JSON-like document storage. This structure is highly beneficial in field applications where data can vary across records, such as when storing pothole reports that may or may not include images, GPS metadata, sensor readings, or manual annotations.

In addition to its schema flexibility, MongoDB supports geospatial indexing through 2d and 2d-sphere indexes, making it capable of executing complex spatial queries. This is especially relevant in mapping pothole locations or querying field events within a geographic radius. Its native replication and sharding mechanisms support fault tolerance and horizontal scalability, ensuring high availability and consistent performance across distributed networks. For offline resilience, lightweight MongoDB instances can be deployed on edge devices, capturing data locally and synchronizing with central servers when a connection is restored.

However, MongoDB has limitations. It operates primarily with eventual consistency, which may lead to synchronization challenges in scenarios requiring strict transactional guarantees. Support for complex JOIN operations is limited compared to traditional RDBMS solutions, and GridFS, MongoDB’s mechanism for storing large binary files such as images, requires additional configuration and resources. Furthermore, MongoDB's use of the Server Side Public License (SSPL) introduces legal and compliance concerns for agencies bound by open standards or procurement restrictions [46].

PostgreSQL

PostgreSQL is a robust, open-source relational database system known for its adherence to SQL standards, ACID-compliant transactions, and extensibility. For projects involving government oversight and infrastructure monitoring, PostgreSQL offers a structured and reliable solution for data integrity and regulatory compliance. One of its standout capabilities is PostGIS, an extension that brings a suite of spatial functions, such as coordinate

transformations, distance computations, and polygon based mapping capabilities well-suited for applications that depend on precise geolocation and route tracking [47].

Data stored in PostgreSQL is easily normalized, and the relational model allows for clearly defined constraints and dependencies, reducing data redundancy and improving integrity. Its transactional guarantees ensure that updates to geographic and photographic data are preserved without risk of loss or duplication. PostgreSQL also allows the integration of multimedia content either directly via BLOBs or indirectly through URI references to cloud-based storage systems.

PostgreSQL like any other software has its drawbacks. To support replication and partitioning, scaling write-heavy workloads horizontally requires careful architectural planning. Unlike MongoDB or SQLite, PostgreSQL does not natively support embedded deployments, which complicates its use in offline mobile environments. In such cases, a synchronization layer must be added between the device and the server, increasing development complexity.

Oracle Database

Oracle Database is a leading enterprise-grade RDBMS that has long been used in government, finance, and defense applications where security, availability, and data fidelity are paramount. Its rich feature set includes support for Oracle Spatial and Graph, an integrated suite of spatial functions and tools for geolocation, mapping, and routing. This functionality allows the database to manage extensive geospatial data with a high degree of accuracy and performance [48].

Oracle's architecture supports Real Application Clusters (RAC), Data Guard, and automatic failover, providing unmatched redundancy and uptime. These capabilities make it an excellent candidate for centralized data storage in critical infrastructure systems. Moreover, it supports multimedia data handling via BFILE references and SecureFiles for efficient image storage and retrieval. Advanced access control, auditing, and encryption further reinforce Oracle’s suitability for projects involving sensitive civic data.

However, Oracle Database is resource-intensive and requires specialized personnel for configuration and administration. Its licensing costs are among the highest in the industry, making it a practical choice only for projects with significant budget allocation and long-term maintenance planning. Oracle also lacks a lightweight deployment model, making it impractical for edge or mobile applications where embedded capabilities are required.

Microsoft SQL Server

Microsoft SQL Server is a relational database management system developed by Microsoft and widely adopted in government, enterprise, and institutional IT environments. It provides comprehensive support for structured data, transactional processing, and business intelligence, while offering enterprise-level features such as role-based security, data auditing, and automatic backup. SQL Server's robust architecture and long-term vendor support make it a strong option for centralized systems requiring stability and compliance.

One of SQL Server's significant capabilities is its integration of geospatial functions through SQL Server Spatial. This allows the database to store, index, and query geographic and geometric data types natively, enabling it to handle GIS-related workloads, including mapping and proximity analysis, which are central to pothole detection and infrastructure mapping tasks. It also supports rich data types, including XML, JSON, and BLOBs, allowing the integration of photos and unstructured sensor data into centralized workflows.

SQL Server can scale vertically on powerful hardware or be deployed in distributed high-availability environments using Always On Availability Groups. Its integration with tools like Microsoft Power BI and ArcGIS Server further enhances its usability in government data pipelines, particularly where spatial data visualization and reporting are important.

Despite its strengths, SQL Server is a commercial product with licensing costs that may be prohibitive for smaller projects or departments. It is also more tightly integrated with the Windows Server ecosystem, which may be limiting in heterogeneous environments where Linux-based deployments are preferred. Additionally, while SQL Server supports many enterprise needs, it is not suitable for edge deployments or embedded mobile applications and is best positioned as a centralized backend.

SQLite

SQLite is a lightweight, embedded relational database widely used in mobile and edge-computing environments. Its minimal setup, file-based storage, and ACID-compliance make it ideal for scenarios where reliable data logging is needed without requiring a continuous network connection. In field applications such as pothole detection, SQLite can store text, GPS coordinates, timestamps, and even images as BLOBs, enabling a comprehensive record to be captured offline and transmitted later [49].

One of SQLite’s greatest advantages is its simplicity. With no need for a server or user authentication, it integrates directly into mobile applications and provides immediate access to the data layer. It is also supported by many programming languages and mobile SDKs, allowing for consistent deployment

across devices. For basic geospatial capability, extensions like SpatiaLite can be used to enhance SQLite with spatial data functions, though with more limited coverage compared to full GIS systems.

Despite these benefits, SQLite does not support concurrent writes effectively, which limits its use in high-throughput environments. It also lacks many enterprise features such as replication, clustering, and user-level security controls. While well-suited for edge use, SQLite is not intended as a primary data store for large-scale, multi-user systems.

Table 3.3.7 *Comparative Table of Databases*

Database	Type	Offline Capability	Geospatial Support	Image/ Binary Support	Scalability	Licensing	Mobile Use
MongoDB	NoSQL	Yes	Yes (2d/2dsphere)	Yes (GridFS)	Horizontal (Sharding)	Free /Paid	Yes
PostgreSQL	Relational (SQL)	Limited	Yes (PostGIS)	Yes (BLOBs/ URI)	Vertical/Partitioned	Free	Limited
Oracle Database	Relational (SQL)	No	Yes (Spatial & Graph)	Yes (BFILE, SecureFiles)	Horizontal (RAC)	Paid	No
Microsoft SQL Server	Relational (SQL)	No	Yes (SQL Server Spatial)	Yes (BLOB, JSON)	Vertical/Clustered	Paid	No
SQLite	Embedded Relational (SQL)	Yes (Native)	Limited (via SpatiaLite)	Yes (BLOB)	None (Single File)	Free	Yes

3.3.5 Evaluation of GPS Frameworks for Mobile Application Integration

The mobile application design requires seamless integration of GPS capabilities for both real-time location tracking and historical route reconstruction (i.e., backtracking) to ensure reliability. The GPS framework must also consider compatibility with mapping services, resource usage, offline functionality, and the ability to interface with embedded hardware over Bluetooth. To ensure that the mobile application met the requirements for this project, we researched many GPS technologies, including those that are already available, such as Google, Leaflet, Waze, and other SDKs. A key consideration in this selection process is

the ability to support persistent pins. Because multiple users and interfaces may access location data, it is necessary that pins representing important locations—such as pothole detections, remain visible and accessible across sessions. This section evaluates viable GPS service options for mobile platforms, referencing software development studies and field applications to guide design decisions.

Native GPS APIs

Both Android and iOS offer native APIs that provide direct access to the device's GPS hardware. Android uses the Fused Location Provider (FLP), which intelligently blends data from GPS, Wi-Fi, and cellular networks to maintain accurate positioning while preserving battery life. On iOS, the CoreLocation framework supports similar functionality through assisted GPS and motion sensors.

As noted in [51] highlights that native APIs generally provide the best results for applications needing precise and continuous location updates, especially outdoors. These APIs are well-suited for logging position over time, enabling effective backtracking when combined with time stamps. However, developers are responsible for storing and rendering past coordinates, as there is no built-in route visualization. To improve timing consistency, one-time time synchronization from the phone to the microcontroller over Bluetooth can help align logged GPS events with MCU-side data—a method supported by the findings in [56].

One advantage of using native GPS APIs is their high precision and responsiveness. While they do not support visual pins or map overlays directly, developers can create persistent pin functionality by storing GPS coordinates locally or remotely and reapplying them on application launch. This approach ensures continuity across user sessions when combined with external map rendering libraries such as Mapbox, Leaflet, or Google Maps [58]. They avoid external dependencies or usage-based costs and allow developers to control how location data is used and stored. However, they also require additional development effort to implement mapping features or visualize routes, as these functions are not provided out of the box.

Google Maps SDK

The Google Maps SDK provides a comprehensive mapping interface, built-in GPS tracking, and integration with the Directions and Places APIs. It also supports drawing polylines that represent travel paths, making it easier to implement backtracking. The SDK ties into the Fused Location Provider, benefiting from Google's infrastructure for location accuracy.

As noted in [52], Google Maps' familiar UI and reliable tracking tools improve user engagement and reduce development time. Backtracking is

well-supported through polylines, and time-stamped location points can be displayed with markers or animated paths. Time alignment for microcontroller logging can also be facilitated by sending a timestamp during the initial Bluetooth handshake, allowing for synchronized route playback.

This SDK's advantages include its strong visual features, well-maintained documentation, and real-time update capabilities. It supports persistent markers by allowing developers to store and later restore coordinate data, ensuring that important locations remain visible even after the session ends [59]. However, it incurs usage fees for applications that exceed the free tier and generally requires a stable internet connection to function fully.

Mapbox SDK

Mapbox offers a flexible mapping platform with offline support, customizable map styles, and robust GPS tracking. It uses native device sensors and allows drawing user paths using its route rendering features. Its ability to cache map tiles locally gives it an edge for applications in areas with unreliable connectivity.

As discussed in [53], Mapbox performs well in scenarios needing offline operation and stylized visualizations. Developers can log timestamped coordinates and plot them for backtracking, and the platform's support for vector tiles enables smoother rendering of movement paths. Like other SDKs, initial time synchronization via Bluetooth ensures event consistency between mobile and embedded systems.

Mapbox works effectively in offline contexts and supports a high degree of design flexibility for the visual presentation of data. It also supports persistent markers through its annotation and data source layers. Developers can serialize pin positions and reload them using local or cloud-based storage, which is critical for maintaining location continuity across user sessions [60]. While it offers deep control, it may require more configuration for advanced services like address lookup or geocoding.

OpenStreetMap and MapLibre

OpenStreetMap (OSM), paired with SDKs like MapLibre or OSMDroid, allows developers to integrate open-source mapping into mobile apps. These tools can render maps, track user movement, and store travel history for backtracking. While they lack some proprietary features, they offer full control and low overhead.

As shown in [54], OSM-based tools are especially useful in constrained environments, where cost and offline access are important. Backtracking is supported through manual plotting of historical location points. Although they do not offer direct support for timestamp synchronization, implementing a one-time

time exchange from the app to the MCU at Bluetooth connection remains effective and is simple to integrate.

OSM-based solutions are attractive due to their cost-free nature, offline readiness, and adaptability. Persistent pins can be implemented by saving and restoring marker coordinates using open libraries like MapLibre or OSMDroid, which support dynamic marker layers [61]. However, they may require developers to handle more of the routing and interface work manually, and the user interface may not be as polished as commercial offerings.

Leaflet

Leaflet is a lightweight JavaScript-based open-source library designed primarily for mobile and web map rendering. Although originally web-focused, it can be used in mobile applications through embedded web views or hybrid frameworks such as React Native or Capacitor. Leaflet leverages raster tile servers, including OpenStreetMap, and allows developers to implement markers, layers, and polylines for route visualization and backtracking.

Leaflet's open architecture supports full control over time-stamped route plotting. With proper implementation, it can log historical paths and replay routes using dynamic polyline updates. While Leaflet does not natively interact with the device’s GPS sensor, it can ingest location data supplied by native APIs, making it flexible for cross-platform use. Bluetooth-based timestamp synchronization, similar to the method proposed in [56], can be implemented at the application layer.

Leaflet’s strengths lie in its lightweight nature, strong developer community, and full compatibility with open-source mapping data. It supports persistent markers through manual storage and recovery of pin coordinates, enabling pins to be rendered consistently across sessions [62]. While Leaflet is effective for hybrid or web-based mobile applications, the reliance on WebView wrappers for mobile deployment may introduce limitations in performance and native integration when compared to fully native SDKs. However, the reliance on WebView wrappers for mobile deployment may introduce limitations in performance and native integration when compared to fully native SDKs.

Waze SDK and Intent-Based Navigation

Waze provides a Transport SDK for select partners that allows applications to trigger navigation in the Waze app. It does not offer map embedding or real-time location updates within the app itself. Developers can only send destination coordinates and receive basic feedback such as estimated time of arrival.

As reported in [55], this model is simple to implement but limits the developer’s ability to manage location tracking or backtracking. Since Waze does not expose its location stream, applications cannot visualize previous paths or

associate time data with specific route segments. It is also incompatible with syncing timestamped data to an MCU.

Although easily integrated, the Waze SDK offers limited flexibility and is not suited for use cases that require in-app tracking, route visualization, or synchronization of movement data with embedded systems. Since it does not allow adding custom markers or accessing in-app location streams, it also lacks support for persistent pins, which makes it incompatible with the requirements of this project.

Table 3.3.8 *GPS SDK Comparison*

Technology	Real-time Tracking	Backtracking Support	Offline Support	Persistent Pins	Map Visualization	Cost
Native GPS APIs	Yes	Manual	Yes	Yes (via storage)	No (requires SDK)	Free
Google Maps SDK	Yes	Yes	Limited	Yes	Yes	Free with usage limits
Mapbox SDK	Yes	Yes	Yes	Yes	Yes	Free/Paid
OpenStreetMap (MapLibre/OSMDroid)	Yes	Yes	Yes	Yes	Yes	Free
Leaflet	Yes	Yes	Yes	Yes	Yes (via WebView)	Free
Waze SDK	No	No	No	No	No	Free

3.3.5 Optimization of Parallel Threads

Python Parallel Processing Methods

The computer vision portion of our design requires an extensive number of parts, not only working in real time but also using a large amount of processing power. It is critical that the organization and optimization of the software are strictly designed to prevent frame loss, bottlenecks, and system crashes. Real-time systems such as the two cameras, decoding, the lightweight detection algorithm, and the ring buffer must be able to run uninterrupted. These systems have only milliseconds to complete a cycle; otherwise, bottlenecks begin to form. Parallelism is critical for allowing each critical process to run independently at the same time. Python and C++ are the most-used languages for computer vision, with C++ offering better parallelism at the cost of much greater complexity. This program, on its own, is already complex enough to heavily favor Python; however, Python does introduce some limitations, and only a few good options exist for working around them.

Pure-Python Multi-Process Pipeline

The main issue with using Python is that its compiler uses a Global Interpreter Lock (GIL), which ensures that only one thread executes Python bytecode at a time within a single process. The GIL makes parallel multithreading impossible for CPU-bound tasks since only one thread can run at a time [64],[65]. To work around this, multiprocessing can be used instead to leverage multiple CPU cores for heavy computing tasks. This works by giving each process its own GIL, preventing CPU-bound work from one process from interfering with another [67]. This effectively dedicates CPU cores to specific processes, as if they are running entirely separate programs with their own set of memory.

This allows for true parallelism of processes, but it also introduces overhead in data movement, memory, and complexity. Since each process now uses its own memory, unlike threads, which share memory, images being passed between processes require pickling and unpickling, which can add 1-5 milliseconds per transfer if not managed well [64],[65],[66]. Each process also has its own Python interpreter, usually consuming around 20-30 MB of RAM, which could push the total RAM usage beyond several hundred megabytes depending on the situation [64],[66]. Starting a new process can also take tens to hundreds of milliseconds on ARM devices, so it is important to use them for long-term processes rather than spawning new ones repeatedly for short-term events [64],[65].

Python with C++ Extensions Pipeline

An alternative option is to add C++ extensions into the Python multiprocessing pipeline for intensive computing routines such as image filtering, infrared image detection, or object detection models. This would use the same multiprocessing design but allows the GIL to be released for selected routines as if they were running in native C++. This will allow parallel threads to run as needed while also allowing multiple processes to run in parallel. The benefit of doing this is that it allows the high-level software to remain in Python, for ease of development, while being able to use the low-level advantages of C++. Some studies have shown that using tools like pybind11 can improve speed up to 2–3 times in real workloads and up to 95 times in synthetic benchmarks when utilizing ARM NEON optimizations [64], [65]. More recent benchmarking efforts show modern binding frameworks can reduce overheads by 2.7 to 4.4 times compared to older solutions like pybind11 and Cython [66].

The biggest downside to using this method is the added complexity of setting up a cross-compilation environment on the Raspberry Pi, managing build scripts, and debugging mixed-language use. Another issue is binding overhead when functions are too small or called too frequently, which can offset performance gains. It is also important to use correct integration since bypassing the GIL can lead to race conditions, segmentation faults, or deadlocks [68],[69].

Python with Numba JIT Acceleration Pipeline

Another possible option is to use Numba's Just-In-time (JIT) compilation to accelerate Python routines, bypassing the GIL for numerical code, while also keeping everything in a single Python process. Numba is a JIT compiler for Python used to speed up numerical code by converting math functions into fast machine code using a Low-Level Virtual Machine [70]. Numba works by compiling Python functions into native machine code at runtime and, while setting "nogil=True", releases the GIL allowing multiple threads to execute heavy loops in parallel on different CPU cores [68],[69]. This would be applied to performance critical routines, like the lightweight laser detection algorithm, allowing for parallel multicore execution without leaving the Python process.

Advantages of using this method include low overhead, sub-millisecond handoff between threads, true parallelism, and simplification for embedded system deployment. The main disadvantages of this method include adding possibly hundreds of milliseconds to the startup of the program, not supporting all complex Python logic, slowdowns when certain loops are applied, and higher debugging complexity. Overall, this approach reduces process overhead and simplifies deployment, at the cost of JIT startup time and some limitations in dynamic Python features [66],[68],[69],[70].

Table 3.3.9: *Comparing Parallel Processing Methods for Python*

Category	Pure-Python Multi-Process	Python with C++ Extensions	Python with Numba JIT
CPU Throughput	Fair	Excellent	Good-Excellent
Latency	Fair	Good	Excellent
Memory	Poor	Fair	Excellent
Startup	Fair	Poor	Fair
Code Complexity	Good	Poor	Good
Debugging	Excellent	Fair	Fair
Deployment Ease	Excellent	Poor	Good
Parallelism Type	Multi-process	Both	Multi-thread

Using Python with Numba JIT is overall the most balanced way to achieve the parallelism our software needs. It achieves CPU-bound throughput results almost as good as using C++ extensions without having to worry about the coding complexity. Since it only uses multi-threading, the RAM usage is considerably lower compared to the other options since using multi-processes requires separate memory for each process [66],[67]. This also removes the 1-5

millisecond latency of sending information from one process to another. While the debugging process may be harder than using the pure-python multi-process method, it makes up for it by having very similar code complexity while achieving much better results for memory, latency, and CPU-bound throughput [68]. If needed, C++ extensions can still be added but with the cost of adding more complexity to the program [63].

3.4 Mechanical Part Selection

The success of the Pothole Detection System is reliant on both the stability and protection of sensing components, including a laser and a camera. In order for smooth detection the mechanical housing and mounting design needs to withstand real world conditions such as vibrations, bumps, moisture, temperature changes, and dust, all while maintaining correct alignment during operation.

Sensor Alignment

Within image based sensing systems, improper calibration and misalignment of optical elements can severely affect performance. This includes skewed frames, defocusing, or a reduced field of view all due to deviations in orientation or position. According to the military optical enclosure studies, optical misalignments as small as 0.1 mm can result in target drift and data invalidation in high resolution systems. [75]. In a system like our project, where the camera and laser system are subject to vibrations and shock loads due to being mounted on the car, these constraints become more strict.

In order to resolve these challenges, the mechanical design of the housing unit must suppress six degrees of freedom. This ensures that after being aligned during assembly, the optical elements resist shifts due to vibrations, driving forces, or stress of the housing material.

Vibration Isolation

There’s been a fair amount of research documenting the effects of low frequency, road induced vibrations on mounted systems used for sensing and imaging. These vibrations are usually generated from potholes, uneven roadways, suspension dynamics, and speed variations are the most intense within the 5 to 30 hz range [76].

This is important data to consider when creating the mounts as a natural frequency that overlaps 5-30 HZ can cause resonance, leading to misalignment or calibration drift.

Environmental Sealing and Water-Proofing

Ingress Protection (IP) standards defined by IEC 60529 that are used to classify levels for sealing enclosures from both solids and liquids. IP65 is regarded as the minimum acceptable level for protecting outdoor electronics from water and dust. This rating ensures that there is complete protection from dust and a strong resistance from water splashes, making it suitable for our project, which could be subject to spray from vehicle tires, rain, or sprinklers. IP67 is a higher standard that allows for brief submersion in water, however with our sensor being suspended 5 feet in the air, this standard is not necessary.

Summary

Adjustable and Tool Free Mounting: The mounts must allow for easy and simple adjustment of sensor angle and height to accommodate different vehicle types. Adding tool free mechanisms enable quick installation and repositioning in the field without specialized equipment.

Ruggedness Against Vehicle Dynamics: Mounting systems must withstand vibrations, shocks, and motion caused by driving. Vibration isolating materials and design considerations that avoid resonance in the 5–30 Hz range are essential for stability.

Sealed Enclosures: Housings must meet at least IP65 standards to protect against dust, rain, and splash exposure. Use of gaskets, weatherproof cable ports, and venting membranes helps maintain internal integrity and prevent moisture buildup.

3.4.1 Mechanical Mount and Housing Material Selection

Material Selection: Mounting System

The mounting arm of the pothole detection system must be mounted externally on a vehicle using suction cups and must position the camera and laser housing at a cantilevered length. It is thus subject to dynamic loading, vibration, shock, and thermal cycling. Structural stiffness, low mass, corrosion resistance, and manufacturability were key considerations in material selection.

6061-T6 aluminum was selected due to its excellent strength-to-weight ratio, excellent corrosion resistance, machinability, and relatively low cost. It is a well-proven structural alloy in the automobile and aerospace industries and has sufficient stiffness to resist bending or deflection.

Carbon fiber composite was considered for its extremely high stiffness-to-weight ratio, but was eliminated due to brittleness upon impact, field repair problems, and electromagnetic interference problems.

ABS plastic, though very inexpensive and easy to print, was eliminated for structural mounting parts due to extremely poor stiffness and insufficient UV and heat resistance.

Table 3.4.1: "6061-T6 Aluminum vs Carbon Fiber Composite vs ABS Plastic "

Material	Pros	Cons	Final Selection
6061-T6 Aluminum	Strong, Easy to Machine, Corrosion Resistant	Heavier than composites	Selected
Carbon Fiber Composite	Lightweight & Stiff	Brittle, expensive, difficult to modify	Rejected
ABS Plastic	Cheap, Printable	Weak Under heavy Load	Selected

Material Selection: Housing Enclosure

The enclosure for the laser and camera must possess high resistance to environmental exposure without optical transparency loss. The material must withstand vibration, exclude water and dust, be machinable or 3D printable, and provide for long-term operation.

Polycarbonate (PC) was used due to its high weatherability, hardness, and clarity of light transmission. Used extensively in headlamp lenses and safety glasses, PC maintains clarity and strength at a wide range of temperatures and is resistant to UV yellowing.

Acrylic (PMMA) was also in the running due to its optical properties and being less expensive, but was ruled out due to lower impact resistance and stress-cracking susceptibility. ABS plastic, while being sufficient for interior non-structural components, lacked environmental durability and transparency to be acceptable on the housing window.

Table 3.4.2: "Polycarbonate vs Acrylic vs ABS Plastic "

Material	Pros	Cons	Final Selection
Polycarbonate	Strong, impact resistant, optically clear	More expensive than acrylic	Selected
Acrylic (PMMA)	Optically clear, cheap	Brittle	Rejected

ABS Plastic	Impact resistant, printable	Poor optical clarity	Selected

Material Selection and Testing

Characteristics such as functionality, reliability, and durability for our project is reliant on the mechanical and environmental characteristics of materials used to construct the sensor housing and the mounting arm. These components are the basis of the detection system, mounting the laser and camera physically to endure a series of stresses from vibration, road contact, weather and long term exposure. Because pothole detection is based on reliable optical alignment, any deformation, warping, or cracking would create detection errors, so the selection of proper materials is crucial in the design.

To address these challenges, we conducted a materials science assessment that considered mechanical strength and fatigue resistance. The application of 6061-T6 aluminum for the mounting system and polycarbonate (PC) for the enclosure was decided after a careful comparison of mechanical performance characteristics, including tensile strength, toughness, ductility, and stiffness.

3.4.2 Mechanical Properties and Behavior

Understanding of the mechanical response of the materials when under stress is required in ensuring the pothole detection equipment runs under a number of realistic stresses. Materials will be classified into three stress regimes of deformation: elastic, plastic, and brittle.

1) **Elastic Behavior** is reversible deformation, meaning material will return to its original shape when the loading is reversed. This behavior is a linear function of stress and strain, as given by Hooke's Law:

$$\sigma = E \cdot \varepsilon \qquad [81]$$

σ is stress, ε is strain, E is Young's Modulus

2) **Plastic Behavior** is when a material exceeds its yield point. The material becomes deformed permanently and does not revert to its original configuration, even upon unloading. This is particularly relevant for crash events or vibration exposure occurring in a repetitive nature, where long term exposure would result in sensor misalignment.

3) **Brittle Behavior** is when materials fail with minimal deformation. Acrylic or glass have brittle failure, which can be dangerous in an environment with potential for impact and vibration.

4) **Ductility**, which is the ability of a material to be able to stretch or deform plastically before it breaks. High ductility allows parts to bend or absorb energy without cracking.

5) **Toughness**, which is the overall ability of a material to absorb energy to failure, as reflected in the area under a stress-strain curve. [78]

For this project, 6061-T6 Aluminum possesses good ductility, moderate toughness, and high strength to weight ratio. It is well suited for use in the external mount, which must be resistant to bending with minimal added load to the vehicle. Polycarbonate (PC) has high impact resistance, high optical clarity, and stability with thermal cycling. It is well suited for the housing that protects the laser and camera components.

These properties ensure the mounting system to resist the 5–30 Hz road induced vibration range without entering resonance or structural fatigue, and the housing to resist environmental loading such as road debris impacts, water exposure, or thermal change.

3.4.3 Tensile Testing of Prospective Materials

To ensure that our material choices were made from quantifiable data, we relied on common tensile test data from the ASTM E8 (Metal Tensile Test) and ASTM D638 (Plastic Tensile Test) methods. Tensile testing establishes a material's elastic limit, yield strength, ultimate tensile strength, elongation at break, and modulus of elasticity. All of these are extremely crucial in selecting materials that undergo real loading.

1) 6061-T6 Aluminum possesses the following mechanical properties [79]:

   Young's Modulus: 69 GPa — extremely rigid for structural integrity

   Yield Strength: ~276 MPa — designates beginning of permanent deformation

   Ultimate Tensile Strength: ~310 MPa — maximum stress before failure

   Elongation at Break: ~10–17% — allows for moderate ductility before failure

2) Polycarbonate (PC) possesses the following mechanical properties [79]:

   Young's Modulus: 2.0–2.4 GPa — rigid enough for enclosure walls, but also has flexibility for impact

   Yield Strength: ~60–70 MPa — sufficient to support sealing and compression loads

Ultimate Tensile Strength: ~65 MPa

Elongation at Break: 60–100% — displays extreme ductility, thus shatter-resistance

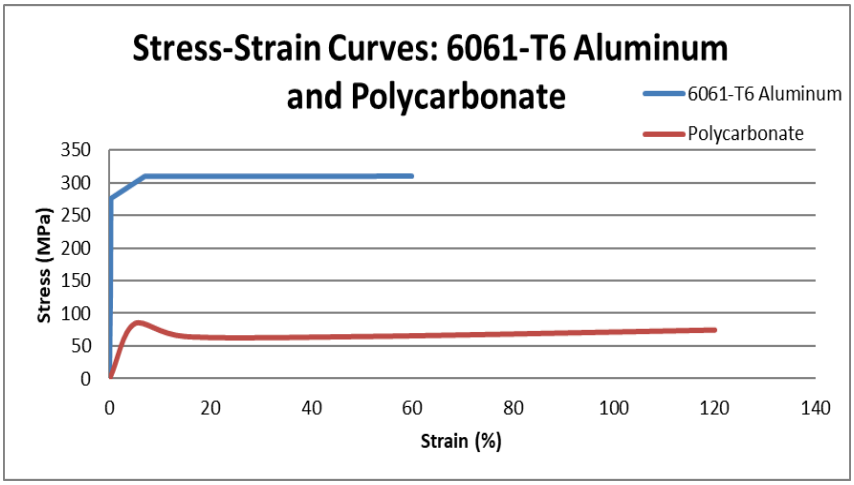


Figure 3.3: Stress-Strain Graph for 6061-T6 Aluminum and Polycarbonate

All these values cater to the system's need for structural rigidity along with flexibility in defined areas. The aluminum ensures that the mount does not fail under load or fatigue due to road vibration, and the polycarbonate casing dissipates impacts without deforming or cracking.

Fatigue Resistance and Cyclic Loading

Because the system is on a moving vehicle with vibrational loading (5–30 Hz range), fatigue is a concern. Testing for fatigue resistance includes ASTM E466 (Metallic fatigue testing) and ASTM D7791 (Plastic tension fatigue testing).[80]

1) 6061-T6 Aluminum is fatigued at ~96 MPa at 500 million cycles and possesses good support for sustained vibration exposure.

2) Polycarbonate, not typically in load carrying, performs well for fatigue at low stress cyclic loading.

Impact Resistance and Fracture Toughness

Polycarbonate was selected for the enclosure window since it possesses excellent impact resistance and high fracture toughness. Unlike acrylic, when impacted, polycarbonate possesses the capacity to absorb a vast amount of energy before shattering. Impact testing includes ASTM D256 (Izod Impact Test) and ASTM D6110 (Charpy Impact Test).

Polycarbonate has the ability to withstand impacts of >850 J/m (Izod), making it an appropriate use in applications where debris or shock incidents can occur.

Compression and Shear Testing

Whereas tensile tests dictate design options, shear and compression properties are equally critical, especially for threaded joints, gaskets, and load transfer between modules.

1) ASTM D732 (Plastic shear strength): Polycarbonate shear strength ~65 MPa.
2) ASTM E9 (Compression testing of metals): Aluminum compressive load can be withstood similar to tensile strength (~300 MPa).

3.4.4 Mechanical Design Calculations and Analysis

To guarantee that the mechanical components of the pothole detection system are reliable under real driving conditions, design calculations were conducted. Calculations include the structural and dynamic stability of the mounting arm and housing enclosure. Because the equipment is mounted and suspended off the back of a vehicle, it will have to undergo cantilevered loading, vibration, and temperature change without compromising alignment or structural integrity. We were able to validate our material selection, optimize size, and ensure that the design will operate safely within known tolerances through engineering analysis of loading, deflection, vibration, and thermal expansion.

Load and Deflection Analysis on Mounting Arm

The mounting arm supports the laser housing and camera at a fixed distance from the surface mass spring system of the car in a cantilevered beam configuration. The arm must accommodate both static weight and dynamic loads caused by road induced motion and vibration. To analyze the structural performance, we calculated the deflection of the arm. The applied load was the total weight of the housing system, estimated at 4.5 pounds (20 N). The arm was assumed to be 0.5 meters long and to be constructed of 6061-T6 aluminum, a material selected for its strength to weight ratio. The moment of inertia (I) of the beam was also estimated at 1.2×10^{-8} m^4 for a rectangular cross section. With Young's Modulus (E) = 69 GPa, we employed the formula for the standard cantilever beam deflection:

$$\delta = \frac{WL^3}{3EI} \qquad [81]$$

When using our values we got:

$$\delta = \frac{20 * (.5)^{3}}{3*69*10^{9}*1.2*10^{-8}} = 1.006\,mm$$

The calculated deflection of the tip was about 1.006 mm, which is considerably less than within acceptable limits for optical alignment. Considering the field of view and pixel size of the camera, this level of deflection results in very low angular misalignment which does not affect performance. The calculation was done to ensure that the arm would be stable and stiff enough under static and dynamic loading without compromising fatigue or bending.

Natural Frequency of Mounting System

Another consideration was that the mounting system itself should not vibrate at frequencies encountered on normal roads. Most vehicle induced vibrations occur between 5–30 Hz, so the natural frequency of the mounting system must be well above this to avoid resonant amplification, leading to structural fatigue or misalignment.

The natural frequency of a mass spring system is given by:

$$f_n = \frac{1}{2\pi}\sqrt{\frac{k}{m}} \quad [82]$$

Where k is stiffness of the arm and m is mounted gear mass. The stiffness k is found using the beam deflection formula:

$$k = \frac{W}{\delta} = \frac{20}{0.001006} = 19880\,N/M \quad [83]$$

Using a weight of 4.5 lbs or 2.04 kg for the housing and arm assembly:

$$f_n = \frac{1}{2\pi}\sqrt{\frac{19880}{2.04}} = 15.7\,Hz$$

This result shows that the natural frequency of the system is just within the critical range, an indication of risk of resonance. Because of this, features such as material optimization, rubber isolators, and damping bushings have been incorporated into design to shift the natural frequency above 30 Hz.

Thermal Expansion of Optical Housing

Since the system is exposed to the outdoors, it must stay aligned and structurally sound even after daily thermal cycling. The housing's optical window, made of polycarbonate plastic, can thermally expand, which will have the potential to misalign the laser or place stresses on the mounting joints.

We used the formula for linear thermal expansion to estimate this:

$$\Delta L = \alpha * L_0 * \Delta T \quad [84]$$

Where: $\alpha = 65 \cdot 10^{-6}$ L=100mm, ΔT=45◦C:

$$\Delta L = 65 \cdot 10^{-6} \cdot 100 \cdot 45 = 0.2925\, mm$$

The expansion was calculated to be less than 0.3 mm, which is not significant to the camera and laser alignment accuracy. Uniform expansion and dimensional stability of polycarbonate are useful for covers that have transparent coverings over housings that are subjected to varying temperatures. This is also in agreement with our use of flexible sealant compounds that can conform to such expansion without failure.

These calculations are important because they turn design ideas into something we can prove will work in the real world. By measuring things like deflection, stiffness, vibration response, and thermal expansion, we made sure the mechanical frame stays strong and aligned during actual use. The results helped us choose the right materials,and shaped the size of parts, the sealing strategy, and how we handled vibration. These choices improve the reliability of the system in outdoor environments and provide the confidence that it will perform safely over time.

3.4.5 Geometry Optimization for Structural Stiffness

The structural stiffness of the mounting arm is an important factor in maintaining proper alignment of the camera and laser throughout operation. Because the sensor housing is mounted in a cantilevered fashion off the rear of the vehicle, small deflections can cause significant angular deviation in the field of view. Misalignment could result in distorted scans, false positives, or detection failure, making geometric stiffness optimization an important design concept.

Cross-Section Design for Flexural Stiffness

In order to obtain deflection resistance from static and dynamic loading, a rectangular aluminium section was designed in the mounting arm. This shape gives an ideal relationship between low weight and high stiffness due to its high area moment of inertia.

The area moment of inertia for a solid rectangular section is:

$$I = \frac{bh^3}{12} \quad [83]$$

Where: *b* = section width & *h* = section height

For a hollow rectangular section, which was used in this design, the formula is:

$$I = \frac{bh^3 - b_i h_i^3}{12} \quad [83]$$

Where: b_i, h_i = inner width and height

This relationship shows that increasing the height of the cross-section exponentially raises bending stiffness, as the term in height is cubed. Because of this, the design was optimized with a taller, rather than wider profile, allowing for stiffening without excessive material use or increased width.

This was balanced with constraints on profile size, weight limits, and manufacturability. A wider cross section would have provided greater torsional stiffness, however it would've reduced stiffness per unit of weight. Our final designed maximized flexural stiffness while maintaining a lightweight form factor

Reinforcements and Adjustable Features

Many design enhancements were considered to preserve structural stiffness and ensure long term reliability. A **t**elescoping extension arm was incorporated, allowing for on site adjustment of arm length based on vehicle geometry. Once the desired extension is obtained, locking collar**s** clamp the tube in place to restore full structural continuity.This feature helps to ensure that the arm remains rigid under various orientations, vibration frequencies, and loading conditions.

3.4.6 Mounting Configuration and Suction Cup Strategy

Five individual suction cups incorporated in this design to balance the mount. There is a mechanical locking lever on each suction cup that, when pressed against a flat vehicle surface, creates and maintains a vacuum seal. The locking mechanism, when locked, provides strong adhesion without adhesives, magnets, or fasteners that are drilled into place.

While no physical testing has yet been conducted for this project, reference values for commercially procured high-strength suction systems suggest that each of the suction cups will resist approximately 100–120 N of pull-off force under ideal circumstances. With five cups being applied in concert, the theoretical holding strength is far greater than 500 N, or over 25 times the static weight of the mounted sensor system (approximately 20 N or 4.5 lbs). This wide margin of safety is required not only to accommodate vertical loading but to resist peel forces and off-axis torques that are experienced in daily driving.

For proper suction cup operation, surface conditions are paramount. For optimum performance there should be:

1) Clean, smooth, non porous surfaces
2) Minimum curvature across the mounting area
3) Ambient temperatures within the cup's specified operating range

In road usage, car surfaces should be clean and dry before mounting. Surface roughness will degrade suction capacity, so smooth surfaces are advised rather than rough surfaces. To maintain long term adhesion, suction status needs to be checked regularly and locking levers need to be engaged.

3.4.6.1 Dynamic Loading Scenarios and Theoretical Performance

Although the test has not yet been performed, the suction mount was selected based on its expected ability to withstand the following conditions:

1) Braking and Acceleration: Deceleration of the vehicle would be most likely to displace the center of mass of the sensor assembly. Five point design loads on the suction cups will ensure stability.
2) Road Vibration: Vibration within the 5–30 Hz frequency range, which is typical of normal driving, has the potential to loosen secured mounts. The mechanical locking aspect of the suction cups counters this with a high friction seal.
3) Wind Drag: At highway speeds, aerodynamic drag can generate considerable rearward force on the housing. Suction adhesion is in a direction perpendicular to the mounting surface, making it resistant to parallel drag forces.

3.4.6.2 Design Advantages Over Permanent Mounts

Compared to adhesive or mechanically mounted systems, this type of suction design has the following practical use advantages.

- Non-destructive: No screwing and drilling, and no vehicle permanent alteration.
- Reusable and adjustable: Capable of removal or relocation from one use to another
- Platform versatile: Suitable for use on a wide variety of vehicles and configurations.

These benefits make it best for use in fleet tests, research studies, or mobile test facilities where modularity and serviceability are of great concern.

3.4.7 Failure Modes and Effect Analysis

Failure Modes and Effects Analysis (FMEA) is an engineering process to identify potential failure points in a system, evaluate the causes and consequences of the failures, and define mitigation strategies to reduce risk. (University of Cambridge) By analyzing each component of a mechanical assembly, FMEA allows engineers to anticipate what can fail, how severe the

effect would be, how likely it would occur, and if it would be detectable before causing major damage. FMEA is used throughout industries such as aerospace, automotive, and manufacturing as it prevents design flaws, warranty issues, and ensures long term system reliability.

A FMEA was critical for our project because of the environmental exposure and real world vibration that the system experiences. Mounted externally on a moving vehicle, the mechanical assembly has to endure road induced shock, moisture, debris, and temperature fluctuations. The laser and camera are subject to misalignment or damage, and their accuracy is crucial to the success of the detection algorithm. Because of this, our team applied the FMEA approach to analyze each component such as the mounting arm, housing enclosure, fasteners, and optical window to determine how each could potentially fail in operation.

For each item, we identified the likely failure modes (cracking, loosening, or seal failure), as well as the causes (material degradation, vibration fatigue, improper assembly). We assessed the effect of each failure on the overall system whether it would result in loss of alignment, water exposure, or system failure and then recommended a method of detection for each case, such as visual inspections, vibration testing, or leak detection procedures. Each failure mode was then graded on three factors: severity (measures how severe the impact of the failure is), occurrence (measures how probable the failure will happen) and detection (measures how easily the failure could be detected before it does damage) These three figures are multiplied together to provide a Risk Priority Number (RPN). The larger the RPN, the more urgent the design revision, testing, or mitigation is required.

The completed analysis revealed the Housing Enclosure and Mounting Arm to be two of the most critical components, with the highest RPN values. This makes sense, as these components supply the structural integrity and environmental protection for the sensing hardware. Failure here would impact the alignment or ruin the electronics, potentially making the system unusable. As a result, we used design enhancements such as redundant seal mechanisms, superior fasteners with thread lockers, and material selection for high corrosion and fatigue strength. These mitigation strategies were taken directly from the FMEA results.

With a rigorous FMEA, we could reveal mechanical design weaknesses and supply justification for specific material and design choices. This task plays a role in achieving the project's reliability goals and in ensuring safe, accurate operation over time.

Table 3.4.1: “FMEA Table”

Component	Failure Mode	Cause	Effect	Severity	Occurrence	Detection	RPN
Mounting Arm	Fracture	Vibration / Impact	Sensor Misalignment/ Detachment	9	4	7	252
Housing Enclosure	Seal Failure	Gasket Degradation	Water Exposure	8	5	4	160
Fasteners	Loosening	Vibration	Component Shift	6	5	4	120
Optical Window	Cracking	Thermal Stress	Image Distortion	6	3	5	90

This FMEA will be revisited after prototype testing to reevaluate risk levels and effectiveness of mitigation strategies.

4. Standards and Design Constraints

4.1 Industry Standards

4.1.1 Electrical Hardware Standards

IPC-2221A

IPC-2221A was developed by the Association Connecting Electronics Industries and is a widely used standard that provides design requirements for printed circuit boards (PCBs) and other forms of component mounting and interconnecting structures. It serves as a standard for making sure the electrical, mechanical, and environment performance of PCBs, focusing on aspects such as conductor spacing, trace width, and thermal management.

Some quantifiable recommendations from IPC-2221A include:

Conductor Spacing

- External Layers: 0.1mm for voltages greater than or equal to 30V
- Internal Layers: 0.13 mm for voltages greater than or equal to 30V

Current Carrying Capacity of Traces:

- A 0.25 mm wide external copper trace supports 0.5A with a 10 degrees Celsius temperature rise
- For a 1A current, trace width should increase to about 0.5 mm under the same conditions

By following IPC-2221A guidelines, designers can make sure that their PCBs will meet industry standards for safety and reliability.

4.1.2 Mechanical Standards

Mechanical standards play an important role in ensuring safety within any engineering project involving physical structures, load bearing components, or environmental exposure. These standards developed by organizations such as the American Society of Mechanical Engineers (ASME), the International Organization for Standardization (ISO), and the American Society for Testing and Materials (ASTM), are pivotal for safe, repeatable, and reliable mechanical systems.

1. **Safety:** Safety is one of the most important drivers behind mechanical standards. These standards exist to ensure that mechanical components are able to withstand stresses, forces, fatigue, and shock without failure. Adhering to the proper safety margins and failure thresholds derived from these standards, helps to prevent and reduce any safety related incidents within industries.

In our project, safety was addressed by referencing the **MIL-STD-810H**, a U.S. military standard that shows how to test equipment for durability, while subject to vibration and mechanical shock. [77] This standard was used when considering the type of material and thickness, vibration damping methods, and the support structure of our mount to ensure that it wouldn't fail when subjected to sudden braking or rough driving conditions. In the case of a collision or high impact vibration, our mount is built to remain both intact and secure to the vehicle preventing damage to surrounding vehicles and pedestrians. This is especially important considering that our system is mounted externally and has direct interaction with motion.

2. **Interoperability:** Mechanical standards show common dimensions, tolerances, hole patterns, and thread sizes that allow parts from various manufacturers to fit and work together simultaneously. This makes it easier to add or switch out parts if needed, promoting flexibility.

Our design followed both standard fastener sizes and bracket mounting patterns based on **ASME B18.2.1** and **ANSI B1.1**, which ensured that replacement parts and components could be sourced without redesign. This essentially allows our design to be used across different vehicle platforms or retrofitted when technology undoubtedly evolves. Following this standard also helps to streamline repair operations, thus reducing downtime.

3. **Quality Assurance:** Standardized specifications help to guarantee the quality of mechanical components by providing criteria for strength, dimensional accuracy, surface finish, and defect limits. They show outlines for testing procedures, which ultimately leads to improved product quality and reduced defects.

To maintain quality and durability within our project, we used data obtained in testing using standards such as **ASTM D638** for tensile strength of plastic and **ASTM B209** for strength of aluminum sheets. This helped to confirm that our polycarbonate housing and aluminum mount would be able to withstand mechanical loads without degradation.

4. **Environmental Protection:** Environmental Protect standards help engineers build systems that are weather and resistant and could function reliably in conditions such as rain or dust.

Because our system is mounted externally to a vehicle, it was pivotal that we met the **IEC 605290 IP65** rating, ensuring that the housing was resistant to rain and water splashes, as well as dust. This also meant making sure all seams, joints, and ports were sealed using gaskets and weatherproof connectors.

4.1.3 Bluetooth Communication Standards

The IEEE 802.15.1 standard originally established the low-level communication rules for Bluetooth, covering how devices connect and transmit data over short distances. It defines the physical and MAC (Media Access Control) layers of a wireless personal area network (WPAN) and was developed to allow multiple devices to communicate efficiently over the 2.4 GHz ISM band using techniques like frequency hopping spread spectrum (FHSS) and Gaussian frequency shift keying (GFSK). These methods were effective in limiting signal overlap and improving the stability of device connections.

This standard introduced a structure for short-range Bluetooth networks, where one device handles coordination and timing, while the others stay in sync with it. The MAC layer helps manage how each device takes its turn when sending data, which prevents overlapping transmissions. Though IEEE no longer actively maintains this standard (marked as withdrawn), its original structure still forms the basis for today’s Bluetooth technologies.

Bluetooth 5.0, the version used in our project, is maintained by the Bluetooth Special Interest Group (SIG). It improves on the original design in several ways. Key enhancements include:

- Faster data rates up to 2 Mbps
- Greater transmission range using coded PHY for improved signal quality
- Support for larger packets (up to 255 bytes), which improves efficiency
- Improved advertising capabilities for faster device discovery and pairing

In the context of our project, Bluetooth 5.0 is used to send sensor and GPS data from the MSP430 microcontroller to a smartphone. Its improved range helps ensure stable communication in mobile environments like a moving vehicle, even when temporary obstacles block direct line-of-sight. The larger packet size and higher speed also reduce the number of transmissions needed, saving power—important for our battery-powered system.

Bluetooth 5.0 keeps a modular architecture with layers like L2CAP (Logical Link Control and Adaptation Protocol), ATT (Attribute Protocol), and GATT (Generic Attribute Profile). These help organize and manage data transmission, ensuring the phone and microcontroller can communicate efficiently and securely. It also includes stronger encryption and privacy protections, which are important for keeping transmitted data secure.

By combining the IEEE's foundational work with Bluetooth SIG’s more recent improvements, our project benefits from a reliable and efficient communication standard that is widely supported by both embedded devices and smartphones

4.2 Design Constraints

4.2.1 Time

The time constraint in the Pothole Detection and Tracking System is one of the most critical challenges in the project. Many of our components rely on real-time detection, and the overall speed of the system plays a major role in its success. The core functionality depends on how quickly the system can detect a road anomaly, confirm it using AI, and transmit that information all while the vehicle is in motion. Because our system is mounted on a moving vehicle, the time constraints become more significant as the vehicle speed increases. The system uses the user’s phone to record the GPS location of each pothole, so if the app doesn’t use GPS backtracking, the data must be processed and sent quickly. This ensures the phone can match the pothole to the correct location when the data is received.

Our performance targets are based on a maximum vehicle speed of 30 miles per hour. At this speed, it's critical for the camera systems to capture frames fast enough to ensure at least one clear frame contains a pothole. For the

infrared camera, which is responsible not only for detection but also for measuring the pothole, it's essential that the entire pothole appears within the laser detection area in at least one frame. Since the laser creates a limited measurement region, we require a higher frame rate from this camera. Based on our calculations, we determined that the infrared camera must operate at a minimum of 30 FPS to guarantee reliable detection and measurement of potholes at 30 mph.

The RGB camera has less demanding frame rate requirements, since it's only used to confirm whether an object is a pothole using object detection. This camera does not rely on the laser region and does not need to capture the full pothole; partial visibility is acceptable. To ensure at least one partial frame includes the pothole, we calculated that the bare minimum is 4.78 FPS at 30 mph. However, for more reliable results and to provide multiple frames for the AI to analyze, we wanted at least 14.34 FPS, so the object appears in about three frames during its visible window.

Another major part of the time constraint is the laser line deviation algorithm. Since the IR camera is running at 30 FPS, the detection logic must process each frame in under 33.3 milliseconds. In a real-world setting, we need to leave some headroom for communication and other system tasks, so we're targeting a processing time of under 30 milliseconds per frame. While the measurement calculations don't need to be real-time, they must be completed not long after the object detection model finishes confirming the pothole.

Although the object detection model is not a real-time system, its processing speed still plays a critical role in the overall timing of the system. Without GPS backtracking on the app, the total time from when the frame is captured to when the detection result is received by the phone becomes extremely important. At a speed of 30 miles per hour, the vehicle moves approximately 44 feet every second, so any delay in processing and transmitting the detection data can cause a significant error in the recorded GPS location of the pothole. Raspberry Pi's limited processing power makes the object detection model the slowest component in the system, and therefore the largest contributor to this delay. If GPS backtracking is not used, the timing constraint becomes stricter, and the accuracy of the pothole location depends heavily on how quickly the model can process the 5–10 buffered RGB frames and send the result. This requires that the model be optimized enough to complete this task within a narrow time window to keep the total delay minimal. We have determined that the maximum time for object detection should be no more than 1 second.

Other factors in our design that contribute to overall system timing include the USB communication speed between the cameras and the Raspberry Pi, the communication speed between the Raspberry Pi and the MSP430, the Bluetooth communication delay between the MSP430 and the user's phone, and the buffer time for the RGB frames. While there are no strict time limits for these individual

components, it is important to minimize their delays as much as possible to reduce the total time it takes for the app to receive the detection data.

In addition to technical timing, the project schedule itself is a major time constraint. As a Senior Design project with strict academic deadlines, delays in any part of the process can have significant consequences. Progress on report writing, hardware testing, firmware development, and software integration is all tied to scheduled due dates. Each student is required to contribute 20 written pages to the project report by the midpoint of the first semester, with the full 150-page report due at the end of that same semester. A prototype demonstration video must also be completed by the end of the first semester, meaning the basic system must be functioning by that time. By the end of the second semester, the entire project must be finalized, including an in-person demonstration and presentation.

Unlike in many other development settings, there is no room for delay in a Senior Design course. To meet these deadlines, the team must stay organized, maintain a clear schedule, and work ahead whenever possible to account for unexpected setbacks. Without strong structure, planning, and collaboration, even a well-executed technical system could fail to meet the course's required milestones.

4.2.2 Cost and Resources

The cost constraint is a major factor in the design and execution of the Pothole Detection and Tracking System. Since this is a Senior Design project, our budget is very limited. The University of Central Florida does not provide us with any financial support, leaving us with only two options. Either securing a sponsorship or covering the costs out of our own pockets. It's also worth noting that one of the key selling points of our design is how much more affordable it is compared to other systems with similar functionality currently on the market. We're further limited by what is commercially available in small quantities, particularly when it comes to specialized technology. While much of the system is designed by hand by our group, some components would either take too long to design given our timeframe or would require manufacturing resources we simply don't have access to. This can limit the level of cost reduction we can implement.

We decided not to get a sponsorship for this project, relying entirely on personal funding. This makes our budget even tighter, since all costs must now be split between the five team members. Our goal is to keep the total project cost under $1200. That way no one must pay more than around $200 total. As mentioned before, what sets our design apart from other available designs is its cost-effectiveness in creating a fully autonomous pothole detection and tracking system. Every part of the design requires extensive research to identify the cheapest component that could still meet our performance requirements.

When working with technologies like cameras, sensors, and lasers, staying within budget can be difficult. These components often come at a high cost, and we constantly must balance quality against affordability while also exploring practical alternatives. For example, LiDAR is one of the most powerful and accurate technologies available for our application. However, even cheaper LiDAR units are extremely expensive, often costing several thousands of dollars. This raises the question of whether the performance benefit is worth the cost, and in our case, it does not. Since there are no cost-effective LiDAR options that meet our needs, we opted for a lower-cost alternative. We decided to pair an infrared laser with a filtered IR camera and add an RGB camera for AI confirmation. This approach allows us to get reasonably close to the detection accuracy of more expensive systems without exceeding our budget.

Resource availability also played a large role in our decision making throughout the design process. Many of the technologies we needed had very specific requirements, and when those features weren't available in cost-effective components, we were often forced to overpay for hardware with more functionality than we needed. One example of this was the RGB camera used in the object detection model. Our design required a camera with a 1920x1080 resolution, a high frame rate, and a field of view between 90 and 100 degrees to match the IR camera. To transmit this data, USB 3.0 was required. However, that combination of specs is rare and limited our options significantly. In the end, we were forced to purchase a much more expensive 3840x2160 camera and downscale the resolution, simply because no other options were available within our constraints. In a more large-scale setting with more time and funding, a custom camera could be sourced or designed to meet only the necessary requirements, reducing the overall cost.

Other hardware costs were also considered throughout the project. While we do have access to more tools at the university than we would at home, we still don't have important resources such as PCB manufacturing. Having this done through a private company tends to be expensive, especially with additional fees like shipping costs and customs taxes. To help keep these costs low, we have options like ordering multiple boards at once, buying from suppliers outside the country, or combining orders with other teams in the class to reduce the per-unit cost. Another example is Bluetooth design. While the most practical and affordable solution would have been to use the Raspberry Pi's built-in Bluetooth, one of the class requirements is that components unrelated to computer vision must be included in our custom PCB. Because of that, we had to look for the cheapest possible Bluetooth design that was still reliable and compatible with our system. When looking at ways to offload computing for our computer vision tasks, we also considered several options. However, most of them were either too expensive or too complicated to integrate, especially considering our soldering limitations in terms of experience. In the end, we had to focus more on performance and reliability than cost, since this is one of the most important parts of the hardware design.

Another major cost-related issue in our project involves training, testing, and prototyping. We can't just simulate hundreds of potholes, and we can't afford to rent out a road or create test potholes on private property. Because of this, we must collect our own training and testing data using potholes we find on public roads. Training an object detection model takes hundreds, sometimes thousands, of labeled pothole images, each with its own unique shape and appearance. While some images can be found online, most of the training data must be collected by us in person, which takes a lot of time and effort. The same problem arises when testing the system. During early testing, we can use artificially made potholes, but for the final version of the project, we need to use actual potholes on real roads, at realistic speeds to ensure it works as intended. This means spending a lot of time driving, testing in multiple locations, and recording accurate results, all of which take up valuable time and resources.

4.2.3 Processing Power

Processing power is a constant constraint that must be considered in nearly every decision we make. One of our project requirements is to have as many components as possible placed directly on the PCB. However, because our project depends heavily on computer vision and multiple cameras and sensors, we have no real options for standalone MCUs. The few that could work would require surface-mount soldering well beyond our current capabilities. Our only realistic option is to use a basic MCU, something like the MSP430, and offload all the processing required for computer vision onto a secondary device that can handle it.

We chose the Raspberry Pi 5 for this role due to its acceptable processing capabilities and variety of I/O ports. While it features a 64-bit 2.4 GHz quad-core Cortex-A76 and 8 GB of RAM, the hardware still comes with its share of limitations. Running two cameras with decoding, preprocessing, a lightweight detection algorithm, and a ring buffer in real time is already a serious challenge. On top of that, the device also needs to handle a computationally heavy object detection model, diameter and depth calculations, SPI encoding, and a range of smaller tasks. It's all technically possible, but only with carefully planned hardware choices, software optimization, multithreading, and a few necessary trade-offs.

Since the Raspberry Pi is only permitted for vision tasks, it's likely outside the scope of the project requirements to include hardware accelerators. Something like the Coral USB Accelerator would be considered second-level offloading and may conflict with the hardware restrictions of the class. While it would give the Pi a significant boost in headroom, it would also narrow our model options and add about $100 to the budget.

One of the biggest computational constraints is the camera and sensor handling. The cameras require a minimum frame rate and resolution to function correctly. The infrared camera needs a much higher frame rate due to its limited

vertical field of view. But increasing frame rate and resolution also increases bandwidth usage. The Raspberry Pi 5 shares a USB bus for all USB 2.0 ports, so if both cameras run over USB 2.0, it'll easily overwhelm the available bandwidth. On the other hand, only one camera should use USB 3.0 due to reported issues with dropped frames when both ports are used simultaneously for cameras.

USB 2.0 offers very limited bandwidth. We can reduce some of the load by using grayscale infrared frames, but even that's pushing it. Affordable camera options that support high-framerate YUY2, with no IR filter, and USB 2.0 compatibility just don't exist. That leaves MJPEG as the only viable option. Encoding images as MJPEG reduces the bandwidth well enough to work over USB 2.0 but decoding them takes significantly more processing power and introduces delay. This becomes a real problem because the image processing, decoding, and lightweight detection algorithm need to finish before the next frame comes in. With the limited processing power available, the only real solution is to isolate the infrared camera, its decoding, and the lightweight detection algorithm into their own dedicated thread.

The RGB camera used for object detection has different needs. It runs at 1920×1080 resolution, captures in color, and only needs half the frame rate of the infrared camera. This still requires a lot of bandwidth, making USB 3.0 the only realistic choice. Fortunately, the RGB camera uses YUY2, which is much lighter on the processor and easy to convert into a usable image. The image processing must happen in real time, but we only run object detection on the RGB images when the IR camera makes a detection. That gives us some breathing room for preprocessing and managing the image buffer, all of which are handled in the same dedicated thread. Using dedicated threads for both camera systems helps minimize bottlenecks, frame drops, and potential system crashes.

The object detection model itself is another major concern when it comes to computing limits. We're very restricted in terms of model size and complexity, especially since the camera systems already take up about half of the Raspberry Pi's processing power. We had to choose the lightest possible model that could still meet our accuracy requirements. Even with the smallest model, it still takes a lot of optimizations to run efficiently. To help with that, we made sure not to overload the model's thread with other heavy tasks. It runs in its own thread, along with any other code that won't get in the way. Everything else, calculations, encoding, SPI transmission, runs on the remaining core.

Finally, the MSP430 MCU has its own severe limitations. It's a very lightweight 16-bit single-core microcontroller with very limited processing capability. Its job is to act as the main control unit for the entire system. Its responsibilities include controlling the laser, handling SPI communication with the Raspberry Pi, and sending measurements and pothole images to the Bluetooth module. Since this is a very light workload, the MSP430 ends up being a suitable choice.

4.3 Mechanical Design Constraints

The Mechanical system must conform to constraints imposed by real world usage, environmental exposure and interdisciplinary team integration. These constraints ensure that the mount and housing not only function under stress, but also remain aligned, safe, and maintainable. Each constraint listed below has been considered and integrated into the project design.

Table 4.3.1: *Mechanical Design Constraints*

Constraints	Requirements	Justifications
Vibration Tolerance	Withstand Vibrations	Prevent sensor misalignment
Environmental Sealing/WaterProofing	IP65 Standard or Higher	Protect against rain, splashes, and dust
Weight Limit	Less than or equal to 5 pounds for housing	Prevent excess load on mount and vehicle exterior
Operating Temperature Range	0C - 45C	Ensure operation in most common temperature range
Serviceability	Modular Design with interchangeable Parts and Easy removal	Helps with ease of repair to reduce complexity

The system is mounted on the exterior of a moving vehicle and is constantly exposed to vibrations, especially when driving over rougher surfaces, potholes, or uneven roadways. Research shows that load frequency road vibrations fall between 5-30 Hz, so the mounting system and housing needs to reflect this. If either of these are not adequately designed, the vibrations can cause sensor misalignment and drift, thus leading to inaccurate detection.

To account for this, vibration damping materials have to be used. Additionally, the design must ensure that the natural frequency of the mount is outside of the typical road vibration range to prevent resonance. This is ensured by strategic material selection and geometry optimization.

4.3.1 Environmental Sealing / Waterproofing

The system will operate in uncontrolled outdoor environments, often experiencing water spray, rain, dust, and debris resulting from tire-kicked up

material. Meeting the IEC 60529 standard, an IP65-rated enclosure offers full protection against dust and protection against water jets from any direction. This level of sealing is necessary to seal off internal electronics inside the case, including the camera, the IR laser, and PCB, all of which are sensitive to moisture and debris. Sealed cable ports, rubber gaskets, silicone sealant, and possibly hydrophobic vent membranes can be employed in order to achieve this rating to allow internal pressure equalization without the entry of internal moisture. Under colder conditions, freeze-thaw cycling also induces condensation and micro-leakage, and thus material compatibility and expansive behavior are issues. Environmental protection maximizes long-term reliability, reduces failure rates, and reduces maintenance needs. Rigorous validation is attainable with lab testing to model rainfall, dust storms, and hose-down cleaning conditions to ensure performance validation.

4.3.2 Weight Limit

The mounting system is externally mounted onto a vehicle and cantilevered off the surface. Excess weight would create a greater bending moment on the mounting arm and add unnecessary stress to the vehicle body, especially when it accelerates, brakes, or vibrates. Keeping the housing at under 5 pounds retains mechanical integrity, reduces reinforcement needs, and averts damage to mounting surfaces. It also ensures that the system can be mounted without engaging structural additions or heavy-duty brackets. Polycarbonate, 6061-T6 aluminum, or reinforced nylon composites are materials with an adequate strength-to-weight ratio. Less weight also facilitates installation and reduces safety concerns involved in handling. If expansion is required (e.g., the addition of sensors), the base system must stay within its quota so that things can remain in equilibrium.

4.3.3 Operating Temperature Range

The temperature range of 0°C to 45°C was selected because it covers the most common environmental conditions the system would be exposed to when operational. Most locations have wide temperature ranges, and components must be able to survive and function within spec through the range. Electronics such as the Raspberry Pi, battery packs, and image sensors must be rated to function within these ranges. The material of the mount and housing cannot warp, crack, or degrade as a result of thermal expansion and contraction. Polycarbonate, for instance, is insensitive to this range and will not yellow or deform easily. Careful thermal engineering, like ventilation or passive heat discharge paths, will be necessary in order to avoid overheating in warm environments. Startup reliability and battery life may be affected in cold environments, so cold-weather testing or preheat logic may be beneficial. Compliance with this temperature range ensures usability in several geographically separated regions and climates.

4.3.4 Serviceability

Field serviceability is critical in a system mounted on fleet vehicles or on city-owned equipment. Over time, elements may wear out, fail, or require upgrading. A modular construction with clearly established mounting points, removable connector cables, and standardized fasteners permits quick repair or substitution of parts without disassembling the entire unit. This reduces downtime and cost and simplifies service and maintenance logistics. Modularity also supports scalable manufacture and enhanced capabilities—new processor or sensor modules can be added without system redesign. Using readily accessible fasteners (e.g., Phillips, Torx), keyed connectors, and color-coded parts makes the system technician-friendly. Enclosures need to allow quick access to interiors without sacrificing environmental sealing on reassembly. Serviceability not only prolongs system life but also facilitates quick deployment and utilization in the actual operating environment where quick turnaround is critical.

5. Pros and Cons of ChatGPT

Artificial Intelligence (AI) refers to the creation of machines and systems that can perform tasks that would normally require a human to do them. These tasks include thinking, learning, problem-solving, and understanding language, all of which are used to enhance human capabilities. In today's world, AI has become a commodity, deeply integrated into daily life and business operations (and school projects), from typical, routine tasks, to highly complex ones.

One of AI's advantages is its ability to process and analyze huge amounts of data quickly and accurately, and can do so at a much faster rate than a human can. This lets AI systems identify patterns, make predictions, and provide insights that influence decision making across a number of industries. Additionally, AI automates repetitive and mundane tasks, which improves efficiency and allows workers to focus on more complex or creative endeavors. Its consistency and precision also reduce the likelihood of human error, which leads to higher quality outcomes.

Well-known examples of AI include ChatGPT by OpenAI, Copilot by Microsoft, and Gemini by Google DeepMind. ChatGPT, launched by OpenAI in November 2022, functions as a chatbot and virtual assistant. It uses machine learning algorithms to process and analyze massive datasets, enabling it to provide real-time, well-crafted responses to user inquiries. Whether its writing an essay, solving math problems, or debugging code, ChatGPT delivers solutions in seconds and can complete tasks that would normally take a human hours to do.

Another notable AI platform is Gemini, formerly known as Bard, developed by Google and released in March 2023. Just like ChatGPT, Gemini can understand and generate human-like text, and respond in real-time. However, Gemini is designed to integrate closely with Google Workspace, which can

improve productivity through suggestions and data analysis within the Google ecosystem.

Although AI has many pros, these systems have their cons as well. The reliability of AI heavily depends on the quality of the data that is used for its training. Biased, outdated, or inaccurate data can lead to poor or misleading results. Furthermore, AI often struggles with understanding nuanced context or handling ambiguous inputs. Models trained in cases outside their training data. Technical issues, such as software bugs or hardware malfunctions can also compromise performance. Lastly, these systems can come with ethical and security concerns, which include potential copyright violations and privacy breaches which pose additional challenges. Addressing these issues is crucial to developing AI technologies that are more reliable, ethical, and trustworthy.

What this chapter hopes to do is look at some of what you might find by posing questions, or prompts to ChatGPT. The design group chose to use ChatGPT over other AI systems because it is the most widely used, and one of the team members has a premium account, which is likely to produce the best solutions or answers possible. In this section the design team is going to pose some prompts to ChatGPT, receive an answer, and comment on what it got right, what it got wrong, what the team liked about the answer the system gave, and what the team disliked about the answer that the system gave. This is important because of the power ChatGPT poses in helping the design team throughout their project.

5.1.1 Case Study 1

Prompt: “Tell me about MCUs”

Response: See Appendix C

Strengths of Response:

This response did a great job of explaining the concept of Microcontroller Units (MCUs) in a clear and specific way. The explanation of an MCU covered all of the essential components such as CPU, RAM, flash memory, I/O ports, and peripherals, without getting too technical, making the information digestible to even a novice, while covering enough that an experienced person can appreciate it. The bullet points, in my opinion, are very helpful and effective with summarizing the typical use cases.

The inclusion of the popular MCUs table is a great addition to this response. It shows concrete examples from major manufacturers and highlights their strengths, which can help readers of all experience levels quickly understand how different families fit into different applications. Similarly, the MCU vs CPU vs Microprocessor is helpful in clearing up the distinction between these

commonly confused terms. This comparison table ties directly to the narrative set by ChatGPT and reinforces the reader's understanding.

Weaknesses of Response:

While ChatGPT's response to the MCU case study was very detailed and informative, there were a few areas where it could have been better. For one, the structure of the content could have flowed more smoothly. The transitions between explaining what an MCU, listing their uses, and then presenting the tables felt a bit short. Adding connecting sentences or brief summaries between these sections would have made the whole explanation feel more cohesive. Also, while the tables were helpful for quick reference, ChatGPT could have included real world examples for each MCU family to make it more relatable and digestible, such as explaining where an MSP could be found or used in a real world application.

5.1.2 Case Study 2

Prompt: "What are the benefits of using switching regulators vs LDO regulators?"

Response: See Appendix C

Strengths of Response:

The response to case study 2 on the benefits of using switching regulators vs using LDO regulators was overall very structured and informative. One of the key strengths was how it broke down the comparison into clear sections, outlining the advantages and disadvantages of each regulator type in a way that was easy to follow. The inclusion of the table was especially helpful for identifying which type of regulator is best for different use cases.

The response also provided real world examples, like powering an ESP32 or an MSP430, which made the information more practical and relatable. Additionally, the explanation provided important design considerations such as efficiency, heat dissipation, and noise, which are relevant for anyone working with embedded systems or power supply design.

Weaknesses of Response:

There were a few areas where the response could have been improved. While the content was very detailed, some parts could have been made more concise to better fit into a technical report without requiring much editing. The response could also have included references to specific datasheets or technical resources in order to strengthen credibility.

Additionally, while the examples were good, adding a small diagram or schematic snippet showing the typical components for an LDO vs a switching

regulator would have provided more visual clarity. Lastly, a brief summary at the end to reinforce the key differences and design trade-offs would have made the section feel more complete and tied everything together.

5.1.3 Case Study 3

Prompt: “What are the pros and cons to use these serial interfaces for the bluetooth module?”

Response: See Appendix C

Strengths of Response:

The response to case study 3 on the pros and cons of using UART, SPI, and I2C for a bluetooth module was very easy to read and well organized. One of its main strengths was how it broke down each interface into clear sections, presenting advantages and disadvantages in well-labeled tables. This made the information easy to navigate and compare. The inclusion of a quick comparison table at the end was very effective in summarizing key features like speed, device support, and ideal use cases. Additionally, the recommendation to use UART for the bluetooth module was very practical and I felt like ChatGPT backed this up with reasoning about typical BLE compatibility. This explanation also had a good mix of technical depth with accessibility, making it useful for readers with different levels of experience.

Weaknesses of Response:

For all of the strengths that the response had, there were some areas where the response could have improved. The response lacked citations from authoritative sources, such as technical datasheets and links to documentation from standards organizations like the IEEE.

6. Hardware Design

6.1 Overall System Design

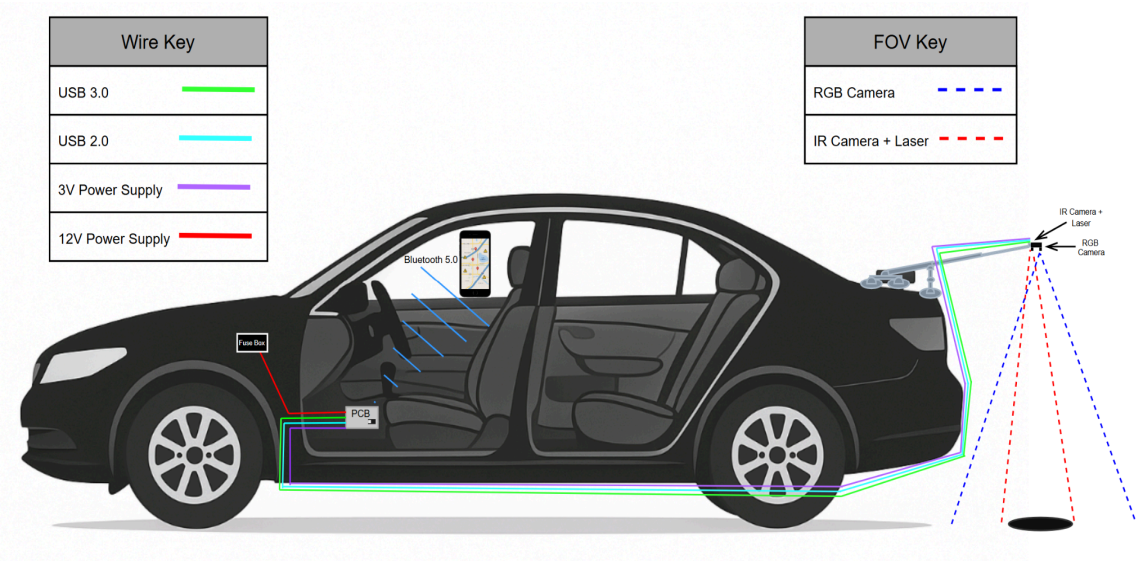


Figure 6.1: Overall System Design Illustration

The overall system design, as shown in the diagram above, is made up of six key components that all work together to ensure the system functions as intended. These components include the PCB housing unit, the user's mobile phone, the 12-volt power supply, the sensor mounting device, the sensor housing unit, and the sensor wire harness. Each one plays an important role in the operation of the system. The PCB housing unit handles processing and connectivity, while the user's phone acts as the interface and data display. Power for the entire system is delivered from the vehicle's 12-volt supply using a fuse tap. The mounting device securely holds the sensor assembly to the vehicle, and the sensor housing itself contains the camera and other sensing components. Finally, the wire harness links everything together, providing both power and data connections from the rear-mounted sensors to the PCB located near the driver's footwell. All of these components were designed to allow for flexibility between different vehicle types, while keeping cost as low as possible.

The PCB housing unit serves as the central hub of the entire system. It is mounted along the side of the driver's footwell, not only for easy access, but also to keep it protected from the elements while ensuring it gets proper ventilation. Inside the housing is the MSP430 microcontroller, a Bluetooth module, voltage regulators, and a Raspberry Pi 5 responsible for computer vision processing. The unit's primary role is to process all data from the sensors and transmit detection results to the user's device via Bluetooth. Power is supplied through a 12-volt

wire routed through the vehicle's firewall, utilizing the factory wire harness grommet to enter the engine bay. There, a fuse tap connects to an existing circuit in the fuse box, allowing the system to safely draw power while remaining protected by the vehicle's original fuse. It's crucial to choose a fuse tied to a circuit that remains on when the vehicle is running, such as daytime running lights, and to avoid any fuses connected to essential safety systems like ABS or airbags. A switch is built into the housing, giving the driver a simple way to turn the system on or off. All voltage levels required for the system are regulated internally within this unit.

The wire harness plays a critical role in both powering the sensor unit and transmitting image data from the cameras back to the PCB housing unit. It consists of three 24 foot wires that include a 3-volt power supply for the laser, an active USB 3.0 repeater cable, and an active USB 2.0 repeater cable. These wires are connected directly to the PCB unit, where the 3V laser power line routes through a voltage regulator, and the two USB cables connect to the exposed USB ports on the Raspberry Pi 5. The harness is fed through the vehicle's firewall using the existing grommet and routed into the engine bay. From there, it travels downward and is securely fastened underneath the vehicle, running along the chassis to the rear of the car. At the back, the harness is routed upward along the mounting system and finally into the sensor unit itself. Because the total length of the harness is 24 feet, far beyond the standard data range for both USB 2.0 and 3.0, it is essential to use active USB repeater cables to maintain signal integrity. Additionally, the voltage regulator must output slightly above 3 volts to compensate for any voltage drop across the long run of wire, ensuring reliable performance of the laser.

The mounting device is a crucial component that ensures the sensor housing unit can be reliably and accurately positioned on the vehicle. One of its key requirements is universal compatibility, as it must be adaptable to a wide range of government vehicles, which can vary significantly in make and model. The design also prioritizes vibration reduction, which is especially important given the amount of movement and road noise generated during normal vehicle operation. This device also has several degrees of freedom, allowing it to be positioned exactly how it is needed. The mount includes a telescoping pole that allows the cameras to be extended far enough behind the vehicle to ensure that the car itself does not appear in any of the captured frames. This not only preserves the accuracy of the sensor data but also maintains consistency across different vehicle setups. The mount's design balances durability, flexibility, and vibration resistance, making it suitable for deployment across diverse vehicle platforms.

The sensor housing unit is specifically designed to securely hold both the infrared and RGB cameras. Since these components are mounted outside the vehicle, protecting them from environmental exposure is critical to system reliability. The housing is waterproof, preventing moisture from entering and damaging the electronics. A key feature of the design is the camera window,

which allows infrared light to pass through, this is essential for pothole detection, as the system relies on infrared imaging for accurate readings. One of the more complex challenges in the design is routing the wires into the housing while still maintaining a waterproof seal, ensuring the system remains protected under all weather conditions.

The final main component of the system is the user's cell phone. The phone runs a custom-built app that connects via Bluetooth to the PCB's onboard module. When a pothole is detected, the MSP430 microcontroller transmits the detection data including the measured area of the pothole, its depth, the time of detection, and an image directly to the phone. The app then uses the phone's GPS to determine the precise location of the vehicle. To compensate for any delay between detection and data reception, the app uses the recorded timestamp to backtrack the phone's GPS data to the exact moment the pothole was detected. This allows it to pinpoint the location accurately. Once compiled, the data is uploaded to our server using the phone's cellular connection. The pothole is then displayed as a pinned marker on a shared map, making it visible to all authorized users.

6.2 Electrical Schematic

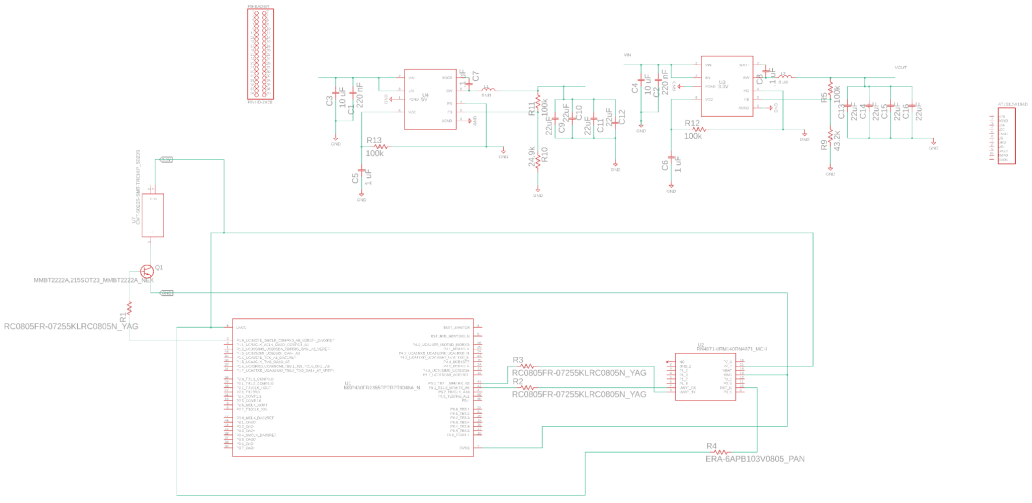


Figure 6.2: Incomplete Electrical Schematic

6.3 Optical Schematic

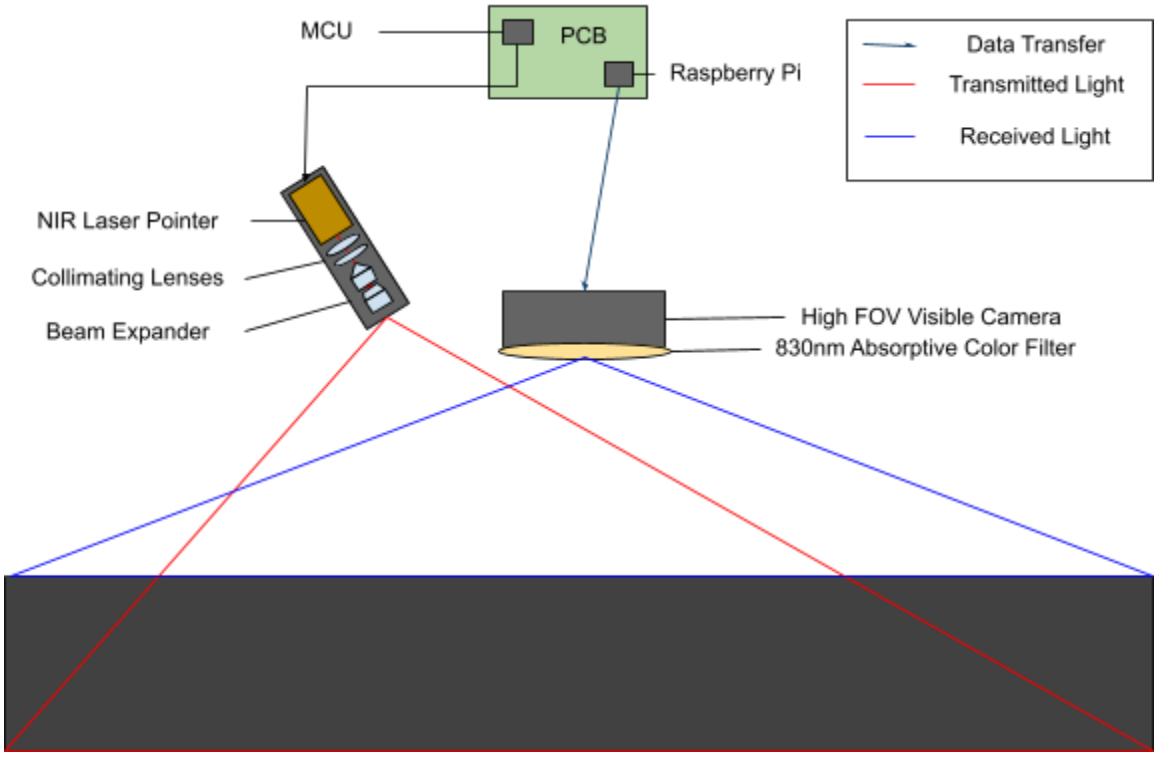


Figure 6.3: Optical Design Schematic

6.4 Mechanical Design

6.4.1 Sensor Housing Design

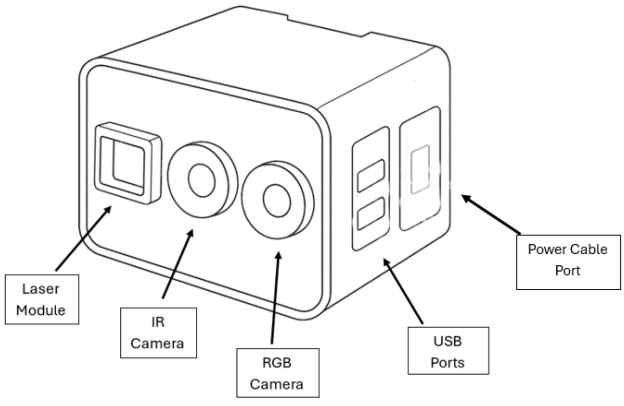


Figure 6.4: Sensor Housing Design Schematic

The Pothole Detection System's housing module is created to securely contain and protect three important sensing components: a laser emitter, an RGB camera, and an infrared (IR) camera, and accommodate the power and data interfaces they need. There are three cutouts on the front of the enclosure, one for each of these sensors. On the left side, there is a laser window positioned in front of the laser emitter and optical lens assembly, where the laser beam can be directed onto the roadway surface. The window includes a clear protective cover, made from polycarbonate, to guard against debris and moisture while being optically clear. Centrally located is the IR camera cutout, providing a view of reflected infrared light patterns from the laser. The opening is covered with an IR transparent material, enabling environmental protection without affecting image quality. To the right of the IR window is the RGB camera cutout, used for high resolution visual imaging, to perform road documentation and AI-based analysis. This opening also features a sealed, clear cover to prevent water exposure while preserving visual data integrity.

The three forward facing openings are arranged symmetrically to preserve each sensor's line of sight without obstruction. Along the right hand side of the enclosure, a panel provides access to two USB ports and a port for the power cable. The USB ports provide power and data transfer to the IR camera, RGB camera, and onboard processing modules such as a Raspberry Pi. Each port is sealed with a rubber gasket or cap to maintain the IP65 environmental rating. Below the ports, the power switch allows manual system control for installation and service. The recessed position and rubber membrane cover are designed to seal out dust and moisture.

Mechanically, the enclosure is constructed of polycarbonate for its balance of impact resistance, UV stability, and relatively light weight. The back of the housing includes a GoPro compatible mount base, allowing universal mounting to the system's telescoping arm or any commercially available mounts. All panel seams, openings, and hardware interfaces are gasketed and compression fitted to allow for IP65 compliance, protecting against exposure to the environment. This configuration meets the system need for optical precision, physical durability, and ease of field service. The layout in general places all principal sensing and power elements in their correct positions while affording a compact weather-sealed compartment that maximizes system life and reliability.

7. Software Design

7.1 Overall Software Design

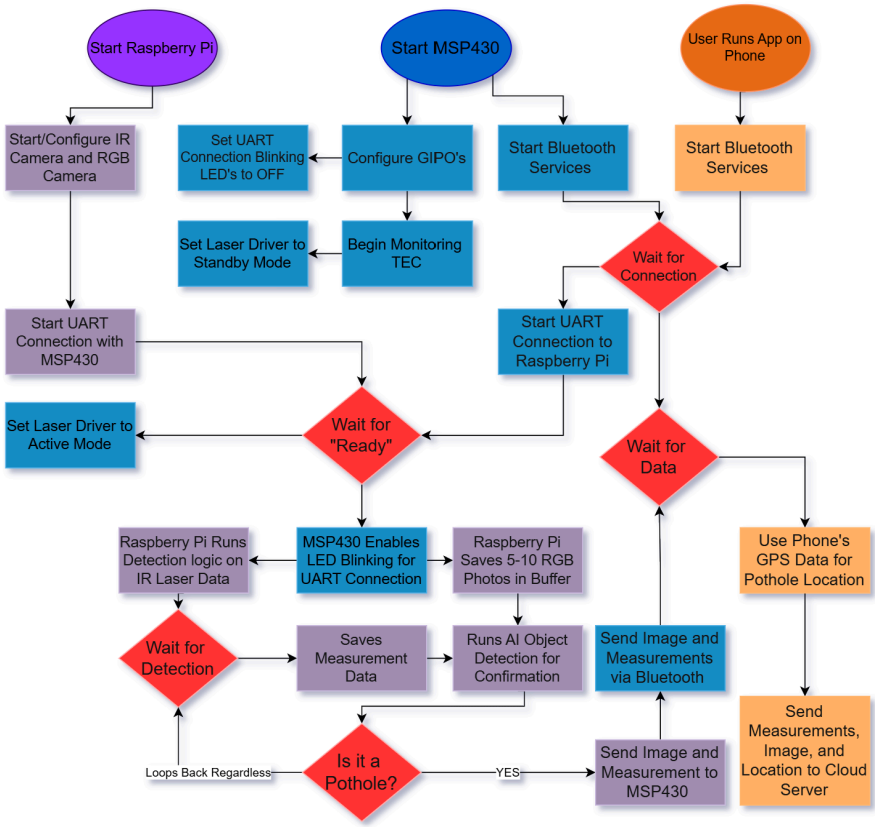


Figure 7.1: Overall Software Flowchart

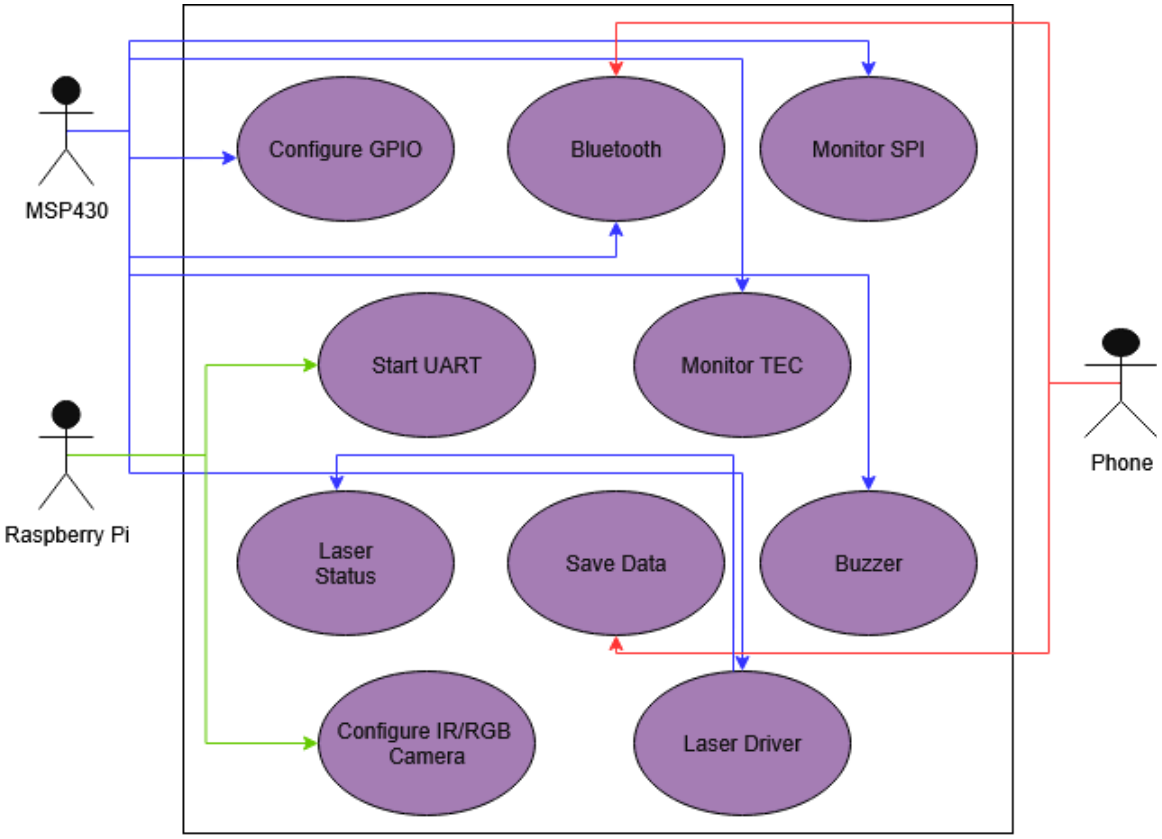


Figure 7.2: Overall Software Use Case Diagram

7.2 Raspberry Pi 5 Software Design

7.2.1 High-Level Module Interaction Diagram

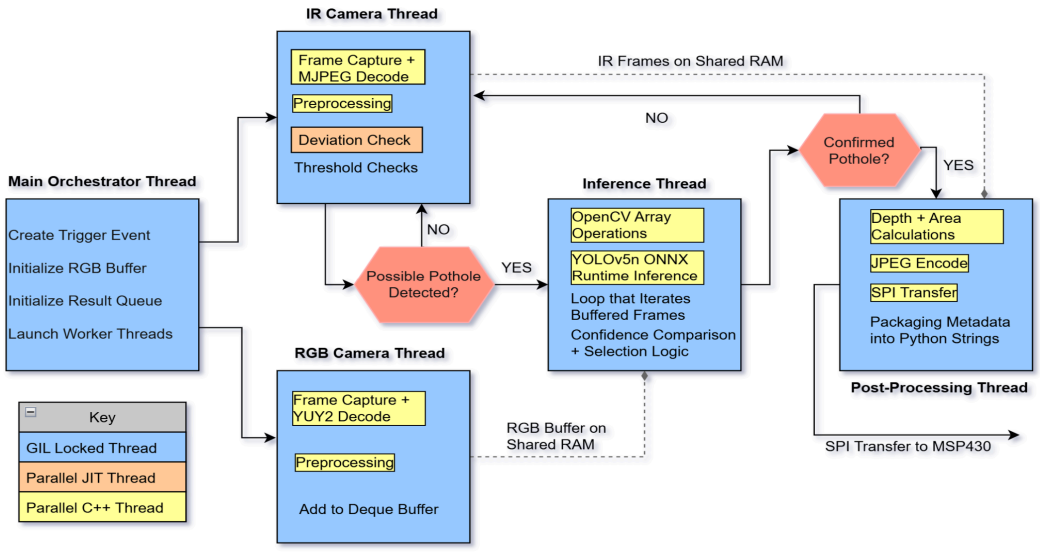


Figure 7.3: High-Level Computer Vision Block Diagram

The program is designed to run on five main threads which include the orchestrator, infrared camera, RGB camera, inference, and post-processing thread. Each of these threads create a group of sub-threads based on their relation and time sensitivity. The five main threads all use the Global Interpreter Lock (GIL) which prevents two CPU-bound threads from running at the same time and prevents true parallelism between the main threads. For the sub-threads grouped inside the main threads, a select number of them drop the GIL allowing true parallel multi-threading. Sub-threads involving OpenCV or NumPy, for example, natively drop the GIL using C++, while the deviation check uses the Just-in-time (JIT) accelerator to achieve this. The organization of the program was carefully designed to optimize the limited computing power of the Raspberry Pi 5.

The orchestrator thread is in charge of initializing the program. It runs once at startup to create and configure all shared data structures including a threading event for the infrared trigger signaling, a collections deque as the RGB frame ring buffer, and a queue for passing inference results to the post-processing thread. Once a SPI connection is established in the post-processing thread, it then spins up the remaining three worker threads, remaining idle until an SPI connection is lost, where it will cleanly terminate all child threads. Since it performs only light Python logic, no heavy loops or blocking calls, it remains entirely GIL-bound.

Dedicated to real-time laser-sheet detection, this thread continuously grabs 1280 x 720 MJPEG frames from the IR camera and each raw frame is immediately converted to grayscale during pre-processing. All three of these calls release the GIL inside OpenCV's C++. For each frame captured, the deviation check is called to quickly look for signs of a pothole without calculating the measurements. It achieves this by thresholding each row, computes subpixel centroids in parallel across all cores, and then calculates an RMS deviation from a flat baseline. This CPU-bound task uses a JIT function, allowing it to drop the GIL. All remaining python code inside the infrared camera thread remains GIL-bound.

The RGB camera thread keeps the inference thread fresh with images to be used if the infrared camera thread flags a possible pothole. It loops at 30 FPS, calling to grab YUY2 frames, and appends each frame into the shared deque. The frame capture, YUY2 decode, and pre-processing all drop the GIL. Since the deque append is a quick Python operation, it momentarily holds the GIL, but it's lightweight enough that frame capture and decoding run at full camera speed.

The inference thread remains idle until the infrared thread flags a possible pothole. When this occurs, the RGB frames currently in the ring buffer are quickly copied to be used. For all frames, a lightweight pre-processing is done before running each frame through the YOLOv5 ONNX inference. If the inference detects potholes in the frames, it compares the confidence scores and bounding boxes dimensions to determine which image it packages into the result queue,

then returns to waiting on the next trigger. The small Python loop that does the selection remains GIL-bound, but the heavy inference work executes in parallel in native code. If a pothole is not detected, it disregards the infrared camera's detection.

Whenever the inference thread enqueues a result, the post-processing thread pulls it off the results queue. First, it uses NumPy to compute the pothole's depth and area from the infrared data image. The RGB image is then compressed to convert the image to a byte array. Finally, it writes the metadata and image bytes out over SPI. All three of these calls use C++, releasing the GIL. Any Python-only steps, like formatting the metadata string, happen under the GIL but have little to no overhead compared to the native operations.

7.3 MSP430 Software Design

7.4 App Software Design

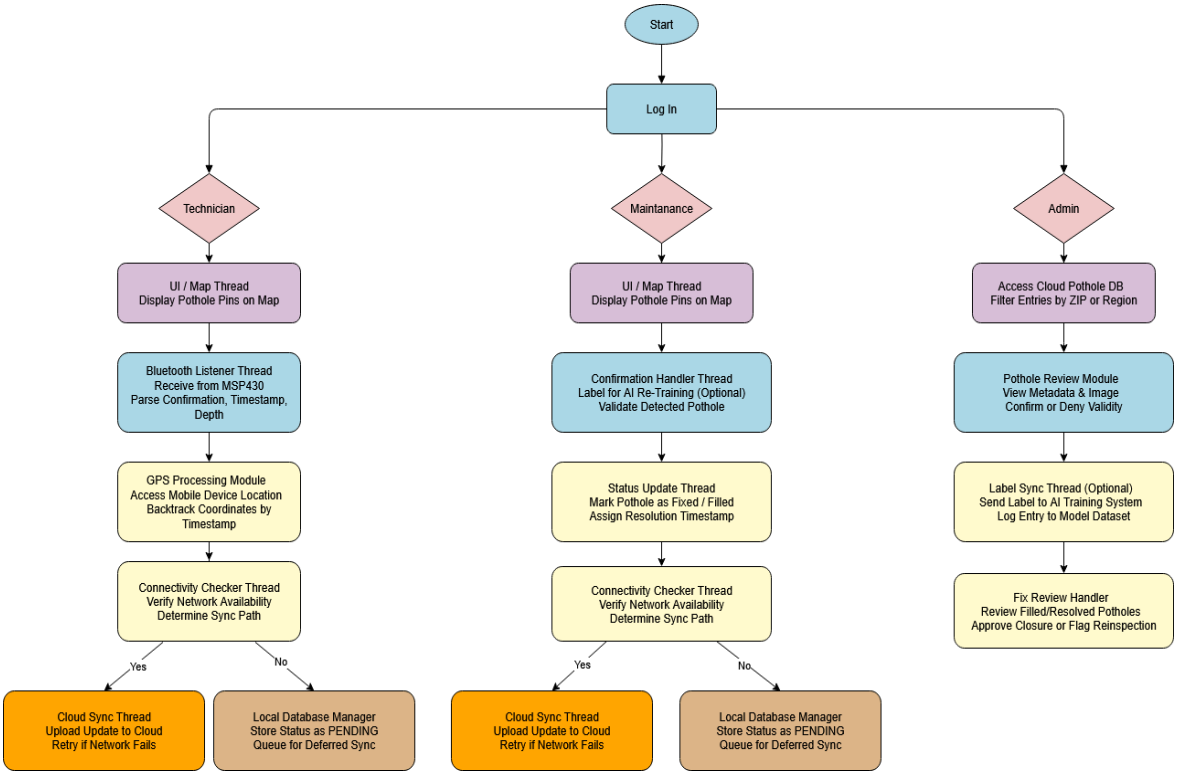


Figure 7.4: Mobile App Block Diagram

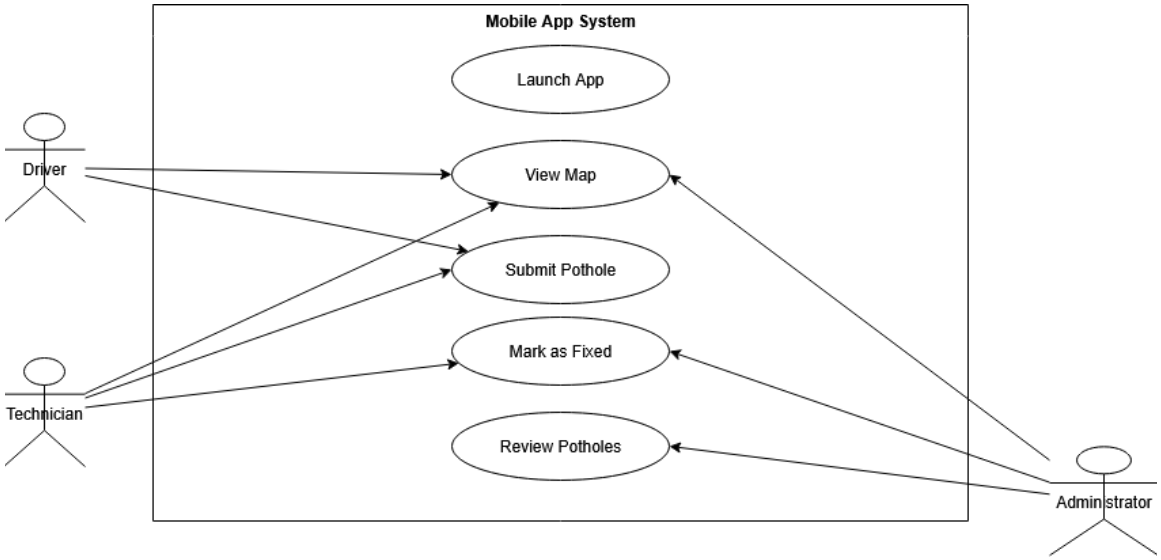


Figure 7.5: Mobile App Use Case Diagram

9. System Testing and Evaluation

9.2 Optoelectronics Feasibility

To confirm the feasibility of the optical design, we conducted an experiment in the CREOL senior design lab that simulated the environment in which our device would be operating. Using a test surface made from multiple foam poster boards glued together painted black, and with a hole cut 1 inch deep into the middle to mimic a pothole on the road. Although our design is built around using infrared light to illuminate the target area, the experiment was conducted with a visible light, helium-neon (HeNe), laser due to its availability and ease of use for testing. The laser light was expanded to our desired size using two perpendicularly aligned Powell lenses and transmitted onto our test surface at an angle of about 55°. For the feasibility experiment, the pothole was imaged head-on using a phone camera (Figure 10.1), and the monochrome IR-sensitive camera that we intend to use in the final design (Figure 10.2).

Figure 9.1: Test surface illuminated by HeNe laser light at an angle.

A ruler was attached to the test area to act as a size reference; however, it did not show up on camera as intended, so the tape used to secure the ruler was used instead, given that it is a standard piece of tape, we were able to determine that it is ¾ in. Using Thorlabs' free camera software, ThorCam™, we were able to measure the number of pixels that corresponded to the ¾ in width of the tape (Figure 10.2)

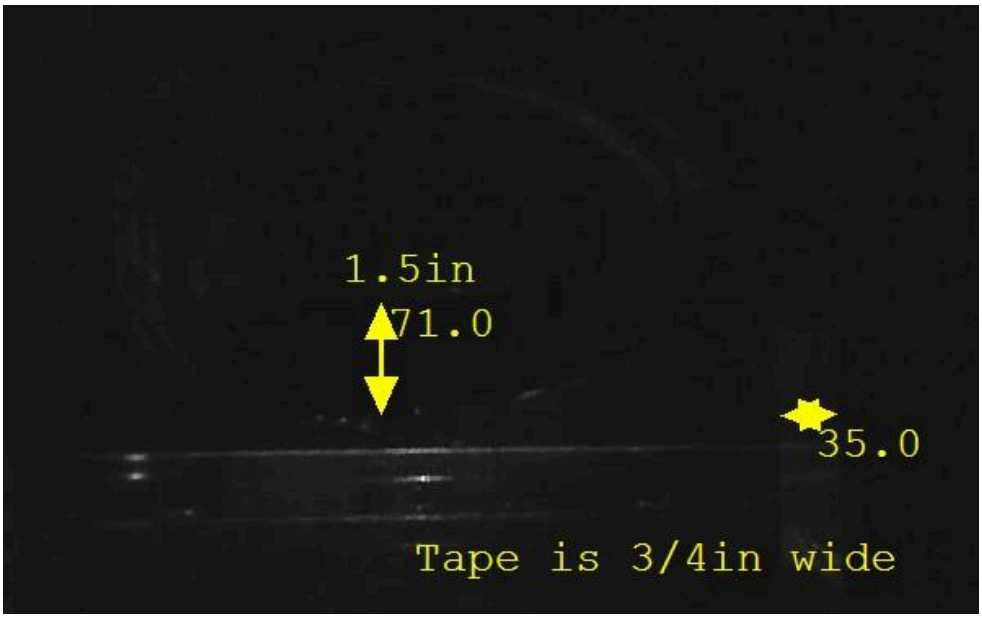


Figure 9.2: Pixel measurements on a monochrome photo of the test area.

By cross-referencing the pixel measurement of the tape to the number of pixels comprising the shadow, we were able to estimate that the shadow is about 1.5 inches in length. Since the incidence angle of the light is known and we have

measured the length of the shadow, we can use Equation 1 to find the depth of the pothole as shown in Figure 10.3 below.

Tan (θ) = $\frac{o}{a}$ Equation 1: Tangent of an angle on a right triangle

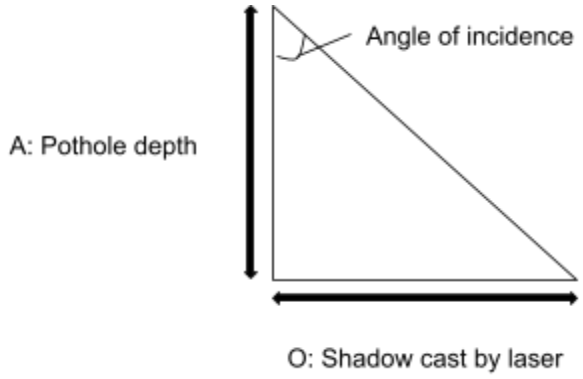


Figure 9.3: Pothole depth measurement using trigonometry.

From these calculations, we were able to estimate that the test pothole is about 1.05 inches. While our calculation is close to the actual depth of 1 inch, it is not perfect. For this experiment, the test area was held by a person, so it is quite likely that the AOI at the time of the image being captured was 1-2 degrees off from the initial measurement, which would explain the discrepancy. Additional testing in a bright environment on an actual road is needed but these results indicate that the design works as intended. The data from our experiment suggests that this optical design will achieve the desired results for our project.

10. Administrative Content

10.1 Budget

The budget for this project is limited to $1200. We set this limit as we will be paying for this project out of pocket and would like to keep costs low. Additionally, a large part of our design is to make the device easy to use and accessible; if the device were too expensive, it would limit its accessibility. As of right now, the budget we are well under budget at around $700, however, this does not account for the possibility that we will need to reiterate the design down the road. Being so far under budget allows us some breathing room on the off chance we need to replace a part with a more expensive alternative.

10.2 Bill of Materials

Table 10.2.1: *Bill of Materials*

Component	Name	Quantity	Price
IR Sensitive Camera	ASIN:B096ZSSCGH	x1	$98.99
RGB Camera USB 3.0	ASIN:B0DBV8VFJZ	x1	$152.28
Housing Unit Materials	TBD	TBD	~$25.00
12V Car Fuse Taps	Nilight - 50038R	x1	$8.09
PCB	TBD	x1	~$40.00
NIR Laser	Big Lasers “IR1” Laser Pointer 808nm 5mW	x1	$99.95
Bandpass Filter	Filter BP Col Hoya RT-830 25.4x2.5mm T	x1	$69.96
Biconvex Lens	LENS DCV 6 X -6 UNCTD TS	x1	$28.09
Plano-Convex Lens	LENS PCX 6 X 18 UNCTD TS	x1	$29.68
Collimating Lens Mounts	OPTIC MOUNT 6.0MM ID	x2	$64.06
45° Fan Powell Lens	B0C68MK6X7	x1	$37.72
110° Fan Powell Lens	B0C68P8M56	x1	$42.57
Raspberry Pi 5	B0CK2FCG1K	x1	$95.99
16-bit MCU	MSP430FR2355TPTR	x1	$2.69
25 foot USB 2.0 extender and repeater Cable	ASIN: B07P8X2MRX	x1	$18.99
25 foot USB 3.0 extender and repeater Cable	ASIN:B081H4N3KQ	x1	$18.99
Bluetooth v5.0 Transceiver Module 2.402 ~ 2.48GHz	150-RN4871-I/RM140-ND	x1	$9.24

Active Buzzer	CMT-5023S-SMT-TR	x1	$1.72
Thermoelectric/ Peltier Mini Module	00411-9J30-20CN	x1	$35.50
On Board Laser Driver	ATLS1.5A104D	x1	$79.00
Total			$957.46

10.3 Milestones and Schedule

10.3.1 Senior Design 1

Table 10.3.1: *Senior Design Schedule*

Week	Description and Dates
1-3	Group formation, Research designs, Write D&C due 5/30
4-5	D&C meeting (6/3), Upload revision to website (6/6), More Research, Start ordering parts, Begin midterm report, Start building optics
6-8	Continue work on midterm report, Start developing code and building CAD designs, Optics Demo(6/26), Test camera framerate, Begin designing PCB
9-11	Finish Midterm (7/7), Test Code with Dev Board + Optics, Midterm meeting (7/9-7/11), Upload midterm revision (7/18), Work on final report
12-13	Work on the final report, mini demo video, and building the optical design. Finish up PCB design and order
13	Finish Final Report, Mini Demo Video, and Optical Design (7/29), Begin prototyping

10.3.2 Senior Design 2

Table 10.3.2: *Senior Design 2 Schedule*

Week	Description and Dates
14-15	School Break
16	Edit a 150-page paper as needed
17-19	Begin building device housing and mounting optics, implement laser diode with TEC and Laser Driver for on-board testing, and begin implementing baseline code
20-22	Connect everything to the PCB and finish building the device housing. Assemble the device
23-24	Begin full system testing, identify problems, and begin solving them
25-29	Lots of testing, make a final demo video, make final changes to the design, and hope for the best
30	Final Committee Presentation

10.4 Table of Work

Table 10.4.1: *Table of Work*

Role	Primary	Secondary	Tertiary
PCB Design	Samuel Welch	Travis Grant	Jose Kostyun
Optics	John Billeci	Travis Grant	Brandon Skervin
Computer Vision	Travis Grant	John Billeci	Jose Kostyun
MCU Programming	Travis Grant	Jose Kostyun	Samuel Welch
Application	Jose Kostyun	Travis Grant	Samuel Welch
Housing Design	Brandon Skervin	Samuel Welch	John Billeci
Mounting Design	Brandon Skervin	Travis Grant	John Billeci

Appendix A | References

[1] A. Anaissi, K. Nguyen, T. Rakotoarivelo, and M. Alamdari, “Smart pothole detection system using vehicle-mounted sensors and machine learning,” *ResearchGate*, Jan. 2019. [Online]. Available: https://www.researchgate.net/publication/330301489_Smart_pothole_detection_system_using_vehicle-mounted_sensors_and_machine_learning

[2] C. Ruseruka, J. Mwakalange, G. Comert, S. Siuhi, F. Ngeni, and Quincy Anderson, “Pothole detection and dimension estimation using in-vehicle sensors,” *Transportation Engineering*, vol. 10, Mar. 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666827024000239

[3] R. Reidy, “Michelin acquires computer vision start-up RoadBotics,” *Traffic Technology Today*, Jul. 12, 2022. [Online]. Available: https://www.traffictechnologytoday.com/news/acquisitions-mergers/michelin-acquires-computer-vision-start-up-roadbotics.html

[4] F. Lambert, “Tesla vehicles are now scanning for potholes and rough roads to help avoid them,” *Electrek*, Jul. 4, 2022. [Online]. Available: https://electrek.co/2022/07/04/tesla-vehicles-scanning-for-potholes-and-rough-roads-help-avoid-them/

[5] Federal Highway Administration, “Road/Intelligent Transportation System (ITS) Maintenance,” *U.S. Department of Transportation*, [Online]. Available: https://ops.fhwa.dot.gov/crowdsourcing/examples/maintenance.cfm

[6] ImageVision, “Pothole Detection Using Computer Vision for Road Quality Management,” *ImageVision*, Feb. 2025. [Online]. Available: https://imagevision.ai/blog/pothole-detection-using-computer-vision-for-road-quality-management/

[7] K. Quinlan, “Memphis piloting AI video tech equipped on city trucks to find, fix infrastructure issues,” *StateScoop*, Dec. 24, 2024. [Online]. Available: https://statescoop.com/memphis-ai-video-tech-equipped-city-trucks-fix-infrastructure-issues/

[8] A. Rubin, “City of San Jose testing use of AI to identify potholes, graffiti and homeless encampments,” *KTVU*, Mar. 26, 2024. [Online]. Available: https://www.ktvu.com/news/city-of-san-jose-testing-use-of-ai-to-identify-potholes-graffiti-and-homeless-encampments

[9] XenomatiX, “XenoTrack | Road Profile LiDAR,” *XenomatiX*, [Online]. Available: https://xenomatix.com/lidar/xenotrack/

[10] G. Gavigan, "LiDAR and AI Used to Detect Potholes," *LiDAR News*, Mar. 28, 2023. [Online]. Available: https://blog.lidarnews.com/lidar-and-ai-used-to-detect-potholes/

[11] A. Jain, “Single-stage detector vs 2-stage detector,” Medium, Feb. 10, 2025. [Online]. Available: https://medium.com/@abhishekjainindore24/single-stage-detector-vs-2-stage-detector-3e540ea81213

[12] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” *in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2015, pp. 91–99.

[13] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, “A survey of deep learning-based object detection,” *IEEE Access*, vol. 7, pp. 128837–128868, 2019, doi: 10.1109/ACCESS.2019.2939201

[14] U. Nepal and H. Eslamiat, “Comparing YOLOv3, YOLOv4 and YOLOv5 for autonomous landing spot detection in faulty UAVs,” *Sensors*, vol. 22, no. 2, p. 464, 2022, doi: 10.3390/s22020464.

[15] O. E. Olorunshola, M. E. Irhebhude, and A. E. Evwiekpaefe, “A comparative study of YOLOv5 and YOLOv7 object detection algorithms,” *Journal of Computing and Social Informatics*, vol. 2, no. 1, pp. 1–9, 2023.

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” *in Proc. Eur. Conf. Comput. Vis. (ECCV)*, Amsterdam, The Netherlands, 2016, pp. 21–37.

[17] M. R. Islam, M. A. Rahman, and Y. M. Jang, “Lightweight deep learning models for real-time object detection on edge devices,” *J. Real-Time Image Process.*, vol. 21, pp. 1–12, 2024.

[18] Ultralytics, “YOLOv5 performance benchmarks,” *Ultralytics Docs*, 2023. [Online]. Available: https://docs.ultralytics.com

[19] Cytron Technologies, “YOLO vs SSD: A detailed comparison for object detection,” *Cytron.io*, Apr. 19, 2022. [Online]. Available: https://www.cytron.io/tutorial/yolo-vs-ssd-a-detailed-comparison-for-object-detection

[20] Dong, Pinliang, and Qi Chen. *Lidar Remote Sensing and applications*. Boca Raton, FL: CRC Press, Taylor & Francis Group, 2018.

[21] Raj, Thinal, Fazida Hanim Hashim, Aqilah Baseri Huddin, Mohd Faisal Ibrahim, and Aini Hussain. "A survey on LiDAR scanning mechanisms." *Electronics 9*, no. 5 (2020): 741.

[22] OpenAI, “ChatGPT”, OpenAI, [Online]. Available: https://www.openai.com. [Accessed 15 June 2025]

[23] S. Ashfaq, M. H. Askari Hemmat, S. Sah, E. Saboori, O. Mastropietro, and A. Hoffman,
“Accelerating deep learning model inference on Arm CPUs with ultra-low bit quantization and runtime,”
in *Proc. Embedded World Conf.*, Nuremberg, Germany, 2022.

[24] X. Shen, P. Dong, L. Lu, Z. Kong, Z. Li, M. Lin, C. Wu, and Y. Wang, “Agile-Quant: Activation-Guided Quantization for Faster Inference of LLMs on the Edge,”
in Proc. AAAI Conf. Artif. Intell. (AAAI), 2024.

[25] N. N. Alajlan and D. M. Ibrahim, “TinyML: Enabling of inference deep learning models on ultra-low-power IoT edge devices for AI applications,”
Micromachines, vol. 13, no. 6, p. 851, May 2022.

[26] R. Lopez Mendez, “Neural network model quantization on mobile,” *Arm Community AI Blog*, Oct. 17, 2023

[27] Szeliski, Richard. *Computer vision: algorithms and applications.*Springer Nature, 2022.

[28] Texas Instruments, *MSP430FR2355 Mixed-Signal Microcontroller Datasheet*, Rev. E, Apr. 2024. [Online]. Available: https://www.ti.com/lit/ds/symlink/msp430fr2355.pdf

[29] Espressif Systems, *ESP32-WROOM-32E & ESP32-WROOM-32UE Datasheet*, v1.6, Apr. 2024. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf

[30] Abdullwahed, S., & Tran, X. D. (2023). Evaluating the Performance of React Native and .NET MAUI for Cross-Platform Mobile Development. Kristianstad University Thesis Repository.

[31] Abu Zahra, H., & Zein, S. (2022). A Systematic Comparison Between Flutter and React Native from an Automation Testing Perspective. Birzeit University.

[32] Biørn-Hansen, A., Grønli, T. M., Ghinea, G., & Yigitbas, E. (2019). An Empirical Study of Cross-Platform Mobile Development in Industry. *Advances in Human-Computer Interaction*. https://doi.org/10.1155/2019/5743892

[33] Ivanović, Z., Kravari, K., & Korošec, P. (2023). A large-scale empirical study on mobile performance: energy, runtime, and code quality in Android apps. *Empirical Software Engineering*. https://doi.org/10.1007/s10664-023-10391-y

[34] Lee, M.-S., Kim, S., & Park, J. (2022). Kotlin, React Native and Flutter: an empirical comparison. In *Proceedings of the 2022 ACM Symposium on Applied Computing*. https://doi.org/10.1145/3590837.3590897

[35] Müller, L.-M., et al. (2022). Understanding the quality and evolution of Android app build systems. *Journal of Software: Evolution and Process*, 27(1), 1–35. https://doi.org/10.1002/smr.2602

[36] Lovrić, L., Fischer, M., Röderer, N., & Wünsch, A. (2023). Evaluation of the Cross-Platform Framework Flutter Using the Example of a Cancer Counse*lling App.* In *Proceedings of the 9th International Conference on Information and Communication Technologies for Ageing Well and e-Health (ICT4AWE)* (pp. 135–142). SCITEPRESS. https://doi.org/10.5220/0011824500003476

[37] Tan, C. W., Lin, C. Y., & Choo, Y. H. (2021). Copilot for Xcode: Exploring AI-Assisted Programming by Prompting Cloud-Based LLMs. Presented at Nanyang Technological University.

[38] Uddin, M., Chen, M., & Lee, S. (2020). An empirical investigation of performance overhead in cross-platform mobile development frameworks. *Empirical Software Engineering*. https://doi.org/10.1007/s10664-020-09827-6

[39] Texas Instruments, *MSP430FR5xx and MSP430FR6xx Family User's Guide*, SLAU367P, May 2013 [Revised June 2024]. [Online]. Available: https://www.ti.com/lit/ug/slau367p/slau367p.pdf

[40] Raytac Corporation, "Downloads – MDBT50Q Series," [Online]. Available: https://www.raytac.com/download/index.php?index_id=43. [Accessed: June 22, 2025].

[41] Microchip Technology Inc., *RN4870/71 Bluetooth Low Energy Module Data Sheet*, DS50002489C, July 2021. [Online]. Available: https://ww1.microchip.com/downloads/aemDocuments/documents/WSG/ProductDocuments/DataSheets/RN4870-71-Bluetooth-Low-Energy-Module-DS50002489.pdf. [Accessed: June 22, 2025].

[42] H. Ahn, T. Chen, N. Alnaasan, A. Shafi, M. Abduljabbar, H. Subramoni, and D. K. Panda,
"Performance characterization of using quantization for DNN inference on edge devices,"
in Proc. 7th IEEE Int. Conf. Fog Edge Computing (ICFEC), May 2023.

[43] P. Biørn-Hansen, F. G. Ghinea, and G. A. M. Jørgensen, "Progressive Web Apps: The Possible Web-native Unifier for Mobile Development," *Proc. 13th Int. Conf. Web Inf. Syst. Technol. (WEBIST)*, pp. 344–351, 2017. Available: https://www.scitepress.org/Papers/2017/63537/63537.pdf

[44] M. S. S. Lingolu and M. K. Dobbala, "A Comprehensive Review of Progressive Web Apps: Bridging the Gap Between Web and Native Experiences," *Int. J. Sci. Res.*, vol. 11, no. 2, pp. 1326–1334, Feb. 2022. Available: https://www.researchgate.net/publication/380908572_A_Comprehensive_Review_of_Progressive_Web_Apps_Bridging_the_Gap_Between_Web_and_Native_Experiences

[45] R. Horn, A. Lahnaoui, E. Reinoso, S. Peng, V. Isakov, T. Islam, and I. Malavolta, "Native vs Web Apps: Comparing the Energy Consumption and Performance of Android Apps and their Web Counterparts," *Proc. 10th Int. Conf. Mobile Software Eng. Syst. (MOBILESoft)*, pp. 44–54, May 2023. Available: DOI 10.1109/MOBILSoft59058.2023.00013

[46] A. Banker, "MongoDB and the SSPL: What You Need to Know," InfoWorld, Nov. 2020. [Online]. Available: https://www.infoworld.com/article/3630187/mongodb-and-the-sspl-what-you-need-to-know.html

[47] R. Obe and L. Hsu, PostGIS in Action, 3rd ed., Manning Publications, 2021.

[48] Oracle Corporation, "Oracle Spatial and Graph Overview," Oracle Technology Network, 2020. [Online]. Available: https://www.oracle.com/database/technologies/spatialandgraph.html

[49] D. Richard Hipp, "SQLite Technical Documentation," SQLite.org, 2023. [Online]. Available: https://www.sqlite.org/docs.html

[50] Microsoft Corporation, "SQL Server Spatial Features Overview," Microsoft Learn, 2024. [Online]. Available: https://learn.microsoft.com/en-us/sql/relational-databases/spatial/spatial-data-sql-server

[51] S. Shankar, A. El-Maleh, and M. N. Qureshi, "Comparative Study of Native and Third-Party Location APIs for Mobile Sensing," International Journal of Mobile Computing and Multimedia Communications, vol. 13, no. 2, pp. 22-35, 2023.

[52] R. Ali and Z. Huang, "Usability and Performance Metrics in Mobile Navigation Using Google Maps SDK," Journal of Mobile Human-Computer Interaction, vol. 10, no. 1, pp. 17-28, 2022.

[53] A. Meixner and A. Papakostas, "Evaluating Mapbox for Mobile GIS Applications with Offline Constraints," IEEE Transactions on Mobile Computing, vol. 21, no. 4, pp. 980-990, 2022.

[54] A. Abd-Elrahman, M. Ghoneim, and R. H. Salem, "OpenStreetMap in Resource-Constrained Mobile GIS: Performance and Trade-offs," Computers, Environment and Urban Systems, vol. 88, 2021.

[55] T. Das, L. B. Morrison, and K. L. Chen, "Integrating Third-Party Navigation Intents into Location-Based Apps: An Empirical Analysis," ACM Transactions on Software Engineering and Methodology, vol. 32, no. 3, 2023.

[56] Y. Huang, D. Sun, and M. K. Ray, "Time Synchronization Techniques in Bluetooth-Enabled IoT Devices," IEEE Internet of Things Journal, vol. 9, no. 6, pp. 4302-4312, 2022.

[57] Leaflet, "Leaflet Mobile Example," LeafletJS. [Online]. Available: https://leafletjs.com/examples/mobile/. [Accessed: Jun. 29, 2025.]

[58] Google Maps Platform, "Add a marker to a map," Google Developers. [Online]. Available: https://developers.google.com/maps/documentation/android-sdk/marker. [Accessed: Jun. 29, 2025.]

[59] Google Maps Platform, "Store and retrieve data," Google Developers. [Online]. Available:

https://developers.google.com/maps/documentation/javascript/examples/marker-simple. [Accessed: Jun. 29, 2025.]

[60] Mapbox, "Add custom markers in Mapbox GL JS," Mapbox Docs. [Online]. Available: https://docs.mapbox.com/mapbox-gl-js/example/custom-marker-icons/. [Accessed: Jun. 29, 2025.]

[61] MapLibre, "Add a marker to a map," MapLibre GL JS Docs. [Online]. Available: https://maplibre.org/maplibre-gl-js-docs/example/add-a-marker/. [Accessed: Jun. 29, 2025.]

[62] Leaflet, "Markers," LeafletJS. [Online]. Available: https://leafletjs.com/reference.html#marker. [Accessed: Jun. 29, 2025.]

[63] Y. Babuji, A. Woodard, Z. Li, D. S. Katz, B. Clifford, R. Kumar, L. Lacinski, R. Chard, J. M. Wozniak, I. Foster, M. Wilde, and K. Chard, “Parsl: Pervasive parallel programming in Python,” *arXiv preprint* arXiv:1905.02158, May 2019.

[64] Z. A. Aziz, D. N. Abdulqader, A. B. Sallow, and H. K. Omer, “Python parallel processing and multiprocessing: A review,” *Academic Journal of Nawroz University*, vol. 10, no. 3, pp. 345–354, 2021.

[65] Z. Aslan and A. Aituov, “Evaluation of concurrency methods in Python for processor-intensive tasks on multicore systems,” *Research Reviews*, vol. 9, no. 2, pp. 34-42, 2025.

[66] A. Jones and B. Lee, “Parallel Processing and Multiprocessing in Python: A Review,” *Journal of High-Performance Computing Systems*, vol. 15, no. 4, pp. 211–224, 2020.

[67] Python Software Foundation, “multiprocessing - Process-based parallelism,” *Python 3.13.5 Documentation*. [Online]. Available: https://docs.python.org/3/library/multiprocessing.html

[68] Numba Developers, “Compiling Python code with @jit,” *Numba User Guide*, 2025. [Online]. Available: https://numba.readthedocs.io/en/stable/user/jit.html

[69] A. Milla and E. Rucci, “Performance Comparison of Python Translators for a Multi-threaded CPU-bound Application,” *arXiv preprint arXiv:2203.08263*, Mar. 2022.

[70] S. Kailasa, T. Wang, L. A. Barba, and T. Betcke, “PyExaFMM: An Exercise in Designing High-Performance Software with Python and Numba,” *arXiv preprint arXiv:2303.08394*, Mar. 2023.

[71] Le, Thithanhnhan, Nam-Tuan Le, and Yeong Min Jang. "Performance of rolling shutter and global shutter camera in optical camera communications." In *2015 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 124-128. IEEE, 2015.

[72] H. Angus Macleod. *Thin-film optical filters.* CRC press, 2010.

[73] BeagleBoard.org Foundation, “BeagleBone AI-64: Design and Specifications”, BeagleBoard Documentation, [Online]. Available: https://docs.beagleboard.org/boards/beaglebone/ai-64/03-design-and-specifications.html#bbai64-design. [Accessed: June 28, 2025].

[74] Raspberry Pi Foundation, “Raspberry Pi 5,” *Raspberry Pi Products*, [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-5/. [Accessed: June 28, 2025].

[75] X. Zhang, et al., “Analysis of Alignment Tolerances in Optical Systems,” Optical Engineering, vol. 56, no. 2, 2017.

[76] N. Hunaidi, “Measurement of Low-Frequency Groundborne Vibrations,” Journal of Vibration and Control, vol. 6, no. 6, pp. 745–758, 2000.

[77] U.S. Department of Defense, “MIL-STD-810H: Environmental Engineering Considerations and Laboratory Tests,” 2019.

[78] ASTM E6-15, “Standard Terminology Relating to Methods of Mechanical Testing,” ASTM International, 2015.

[79] ASM International, *ASM Handbook Volume 2: Properties and Selection: Nonferrous Alloys and Special-Purpose Materials*, ASM International, 1990.

[80] ASTM E466-15, “Standard Practice for Conducting Force Controlled Constant Amplitude Axial Fatigue Tests of Metallic Materials,” ASTM International, 2015.

[81] R. C. Hibbeler, *Mechanics of Materials*, 10th ed. Boston, MA: Pearson, 2016.

[82] F. P. Beer, E. R. Johnston Jr., J. T. DeWolf, and D. F. Mazurek, *Mechanics of Materials*, 7th ed., New York, NY: McGraw-Hill, 2015.

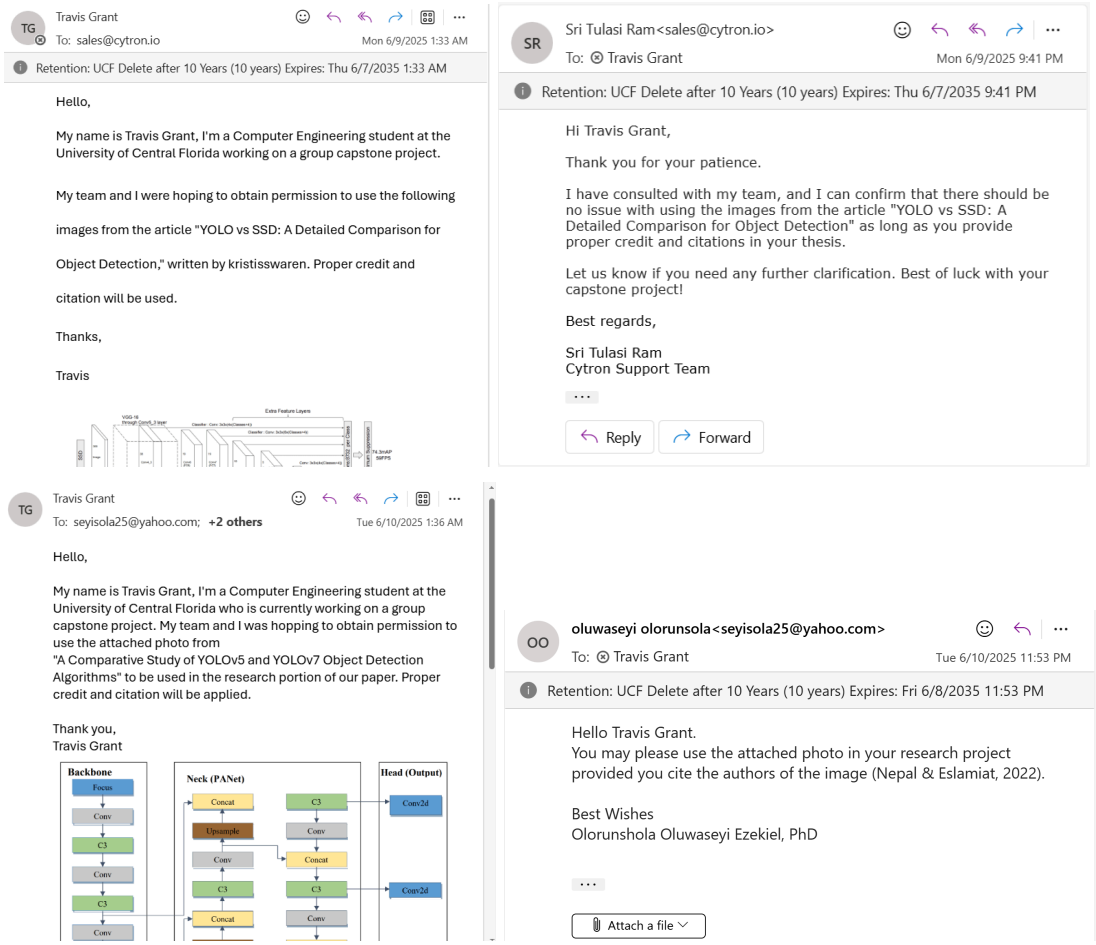
[83] J. M. Gere and B. J. Goodno, *Mechanics of Materials*, 8th ed. Stamford, CT: Cengage Learning, 2012.

[84] [4] F. P. Incropera, D. P. DeWitt, T. L. Bergman, and A. S. Lavine, *Fundamentals of Heat and Mass Transfer*, 7th ed., Wiley, 2011.

[85] IEEE Standards Association, *IEEE Recommended Practice for Powering and Grounding Electronic Equipment (Emerald Book)*, IEEE Standard 1100-2005, 2005. [Online]. Available: https://standards.ieee.org/ieee/1100/1640/

[86] IPC. (1998). *IPC-2221A: Generic Standard on Printed Board Design*. Association Connecting Electronics Industries. Retrieved from https://www-eng.lbl.gov/~shuman/NEXT/CURRENT_DESIGN/TP/MATERIALS/IPC-2221A(L).pdf

Appendix B - Copyright Permission



Travis Grant
To: sales@cytron.io
Mon 6/9/2025 1:33 AM
Retention: UCF Delete after 10 Years (10 years) Expires: Thu 6/7/2035 1:33 AM

Hello,

My name is Travis Grant, I'm a Computer Engineering student at the University of Central Florida working on a group capstone project.

My team and I were hoping to obtain permission to use the following images from the article "YOLO vs SSD: A Detailed Comparison for Object Detection," written by kristisswaren. Proper credit and citation will be used.

Thanks,

Travis

Sri Tulasi Ram<sales@cytron.io>
To: Travis Grant
Mon 6/9/2025 9:41 PM
Retention: UCF Delete after 10 Years (10 years) Expires: Thu 6/7/2035 9:41 PM

Hi Travis Grant,

Thank you for your patience.

I have consulted with my team, and I can confirm that there should be no issue with using the images from the article "YOLO vs SSD: A Detailed Comparison for Object Detection" as long as you provide proper credit and citations in your thesis.

Let us know if you need any further clarification. Best of luck with your capstone project!

Best regards,

Sri Tulasi Ram
Cytron Support Team

Reply Forward

Travis Grant
To: seyisola25@yahoo.com; +2 others
Tue 6/10/2025 1:36 AM

Hello,

My name is Travis Grant, I'm a Computer Engineering student at the University of Central Florida who is currently working on a group capstone project. My team and I was hopping to obtain permission to use the attached photo from
"A Comparative Study of YOLOv5 and YOLOv7 Object Detection Algorithms" to be used in the research portion of our paper. Proper credit and citation will be applied.

Thank you,
Travis Grant

oluwaseyi olorunsola<seyisola25@yahoo.com>
To: Travis Grant
Tue 6/10/2025 11:53 PM
Retention: UCF Delete after 10 Years (10 years) Expires: Fri 6/8/2035 11:53 PM

Hello Travis Grant.
You may please use the attached photo in your research project provided you cite the authors of the image (Nepal & Eslamiat, 2022).

Best Wishes
Olorunshola Oluwaseyi Ezekiel, PhD

Attach a file

Appendix C – ChatGPT prompts and outcomes

5.1.1 Case Study 1

Prompt: “Tell me about MCUs”

Response:

What is an MCU?

A Microcontroller Unit (MCU) is a small, self-contained computer on a single integrated circuit (IC) designed to perform specific tasks in embedded systems. It typically includes:

- A CPU (central processing unit) – processes instructions
- RAM – temporary data storage
- Flash Memory – stores firmware (program code)
- I/O ports – connect to external sensors, actuators, and peripherals
- Timers, ADCs, communication interfaces – enable real-world interaction

What do MCUs Do?

MCUs are used to control hardware in real-time. They're found in thousands of devices you use every day, from smartwatches and appliances to cars and industrial machines.

Typical MCU tasks include:

- Reading sensor data
- Controlling motors, lights, displays
- Managing timing (with interrupts and timers)
- Handling basic decision-making based on inputs

Table 5.1.1: *Examples of Popular MCUs*

Family	Brand	Notes
MSP430	Texas Instruments	Ultra-low power, good for sensing and control
STM32	STMicroelectronics	ARM Cortex-M based, powerful and versatile
ATmega328P	Microchip	Easy for beginners, widely supported
ESP32	Espressif	MCU + Wi-Fi/BLE, great for IoT

Advantages of MCUs

- Low power – ideal for battery-operated devices
- Small size – fits into compact systems
- Real-time control – responds to hardware events with low latency
- Cost-effective – many are under $5
- Highly customizable – code tailored for specific hardware tasks

Limitations of MCUs

- Limited processing power – not suitable for tasks like full image processing or AI interference
- Constrained memory – not good for running large operating systems
- Specialized development tools – often need IDES like STM32Cube, Code Composer, or MPLAB

Table 5.1.2: *MCU vs. CPU vs. Microprocessor*

Component	Function	Use Case
MCU	CPU + memory + peripherals in one	Embedded control (sensors, motors)
CPU	Just the processing unit	General-purpose computing (desktop, server)
Microprocessor	CPU with external memory and I/O	PCs, Raspberry Pi, smartphones

[22]

5.1.2 Case Study 2

Prompt: “What are the benefits of using switching regulators vs LDO regulators?”

Response:

Great question! Both voltage regulators and LDOs (Low Dropout Regulators) serve the purpose of providing a stable output voltage, but they differ in efficiency, complexity, heat dissipation, and use cases.

Voltage Regulator Types

When people say “voltage regulator”, it can refer to two main categories:

1. Linear regulators (e.g., 7805, LM317, LDOs)

2. Switching regulators (e.g., LM2576, buck/boost converters)

An LDO is a type of linear regulator, so let's compare LDOs vs. Switching Regulators since that's usually the decision point in a design.

LDO (Low Dropout Regulator)

Advantages

- Simplicity: Minimal components – just a couple of capacitors
- Low noise: No switching noise; ideal for analog/RF circuits
- Fast transient response
- Cheap and compact: Small footprint, east to design in
- Low dropout: Can regulate with very little headroom (e.g., 3.3V output from 3.6V input)

Disadvantages

- Inefficient at high voltage drops or high current
- Efficiency = Vout / Vin
- Example: 5V to 3.3V - 66% efficient
- Heat dissipation: Excess voltage is burned off as heat
- Limited current: Typically a few hundred mA, some up to ~1.5A

Switching Regulators (Buck, Boost, Buck-Boost)

Advantages

- High efficiency (80-95%+) even with large voltage differences
- Wide input voltage range
- Can step-up (boost) or step-down (buck) the voltage
- Higher current output – several amps or more

Disadvantages

- More complex: Requires inductors, diodes, and control logic
- Switching noise: Can interfere with sensitive analog circuits
- Larger PCB area (especially for through-hole parts like LM2576)
- Slower transient response in some cases

Table 5.1.3: *When to Use LDO or Switching Regulator*

Use Case	Best Option
Small voltage drop (e.g., 3.6V - 3.3V)	LDO
Low current + noise-sensitive analog	LDO

Battery-powered, power-sensitive	Switching Regulator
High voltage drop (e.g., 12V - 3.3V)	Switching Regulator
High current loads (e.g., >1A)	Switching Regulator
Space- and cost-sensitive designs	LDO (if efficiency isn't critical)

Example

If you're powering an ESP32 (3.3V, 300 mA peak) from 12V, then:

- LDO - low efficiency, lots of heat
- Switching buck converter - much more efficient, cooler, better for battery life

But if you're powering a low-power MSP430 from 3.6V down to 3.3V, an LDO is ideal – it's simpler, smaller, and the power loss is negligible.

[22]

5.1.3 Case Study 3

Prompt: "What are the pros and cons to use these serial interfaces for the bluetooth module?"

Response:

Absolutely! Here's a detailed breakdown of the main serial communication interfaces used in embedded systems: UART, SPI, and I2C – with in-depth pros, cons, and ideal use cases.

1. UART (Universal Asynchronous Receiver/Transmitter)

Table 5.1.4: *UART Pros*

Feature	Explanation
Low pin count	Only 2 main wires: TX and RX. Optional RTS/CTS for flow control.
Simple protocol	No clock signal required. Easy to implement in firmware and drivers.
Asynchronous	Devices don’t need to share a clock – great for connecting two devices running independently.
Point-to-point	Great for direct communication between two devices (e.g., MCU - Bluetooth module).
Wide support	Almost every microcontroller and peripheral supports UART natively.
Variable baud rates	Flexible speeds from 300 bps to several Mbps (usually 115200-1M baud).

Table 5.1.5: *UART Cons*

Feature	Explanation
Only 1-to-1	Not inherently multi-device. Can’t share a UART bus like I2C.
No error detection/correction	Basic framing/parity only. CRC must be handled in software.
Not synchronized	Requires accurate baud rate matching.
Lower throughput	Compared to SPI, data rates are lower (usually <1 Mbps reliably)

Best For

- Connecting to BLE modules
- GPS receivers
- Debugging/console output
- Simple, low-speed serial devices

2. SPI (Serial Peripheral Interface)

Table 5.1.6: *SPI Pros*

Feature	Explanation
Very Fast	Data rates can exceed 10-50 Mbps – great for high-speed data transfers.
Full duplex	Simultaneous send and receive (MOSI and MISO).
Simple protocol	No addressing; just select the device with CS (Chip Select) and send bytes.
Flexible	Works well with multiple slaves (if you have extra CS lines).
Precise control	You fully control the clock timing, data edges, and protocol framing.

Table 5.1.7: *SPI Cons*

Feature	Explanation
More pins	Needs at least 4 wires: MOSI, MISO, SCLK, CS. One CS per slave.
No standard protocol	Unlike I2C/UART, you must define the protocol (command formats, timing, etc.)
Shorter cable distances	Susceptible to noise if traces/wires are long or not impedance-controlled.
No inherent device addressing	All devices must have their own dedicated CS line.

Best For

- High-speed devices: flash memory, displays, ADCs
- Short-range, high-bandwidth communication
- Real-time control systems

3. I2C (Inter-Integrated Circuit)

Table 5.1.8: *I2C Pros*

Feature	Explanation
Two wires	Only needs SCL (clock) and SDA (data) regardless of number of devices.
Built-in addressing	Every device has a 7- or 10-bit address. Master can talk to many slaves on the same bus.
Multi-master capable	More than one controller can exist on the bus (with care).
Ack/Nack support	Protocol supports basic acknowledgement and flow control.
Good for sensors	Many sensors (IMUs, temperature, etc.) are designed to use I2C.

Table 5.1.9: *I2C Cons*

Feature	Explanation
Slower	Standard = 100 kHz, Fast = 400 kHz, Fast+ = 1 MHz, High Speed = 3.4 MHz (but not widely used).
Requires pull-up resistors	Needs 4.7kOhm (typical) on SDA and SCL lines for correct operation.
Noisy in longer runs	Not ideal for distances over ~1 ft without shielding or buffering.
One device can block bus	If a slave holds the bus low, the master can’t communicate with others.

Best For

- Connecting multiple low-speed devices (sensors, EEPROMs, RTCs)
- Short-distance board-level communication
- Simpler MCU-to-sensor integration

Quick Comparison Table

Table 5.1.10: “UART vs SPI vs I2C”

Feature	UART	SPI	I2C
Wires	2 (TX/RX)	4+ (MOSI, MISO, SCLK, CS)	2 (SDA, SCL)
Devices Supported	1-to-1	1 master, many slaves (each needs CS)	Many (addressed)
Speed	Low-Medium (~1 Mbps)	High (>10 Mbps)	Low-Medium (~400 kbps)
Full Duplex	Yes	Yes	No
Protocol Simplicity	Very easy	Protocol must be defined	Well-defined
Error Handling	Basic (parity)	Manual	Basic (ACK/NACK)
Ideal Use Case	Simple 1:1 comms	Fast peripherals	Multiple sensors

So, For Your Project:

You’re sending moderate data (GPS, timestamps, maybe flags/images) to a Bluetooth 5.0 module – go with UART unless your module explicitly supports SPI (very rare for BLE modules).

[22]